Un outil graphique pour une interface multi-modes

Helmi Ben Amara

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

UNIVERSITE DE SAINT-ETIENNE

N° D'ORDRE : 75ID

THESE

Présentée par

BEN AMARA Helmi

pour obtenir le titre de

DOCTEUR
DE L'UNIVERSITE DE SAINT-ETIENNE
ET DE L'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE
(Spécialité: Informatique)

UN OUTIL GRAPHIQUE POUR UNE INTERFACE MULTI-MODES

soutenue à SAINT-ETIENNE le 16 Mars 1992

COMPOSITION du JURY:

Monsieur Jean AZEMA
Monsieur Mohamed BEN AHMED
Monsieur Jean-Pierre BRAQUELAIRE
Monsieur Bertrand DAVID
Monsieur Pierre FALZON
Monsieur Bernard PEROCHE

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

UNIVERSITE DE SAINT-ETIENNE

N° D'ORDRE : 75ID

THESE

Présentée par

BEN AMARA Helmi

pour obtenir le titre de

DOCTEUR
DE L'UNIVERSITE DE SAINT-ETIENNE
ET DE L'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE
(Spécialité: Informatique)

UN OUTIL GRAPHIQUE POUR UNE INTERFACE MULTI-MODES

soutenue à SAINT-ETIENNE le 16 Mars 1992

COMPOSITION du JURY:

Monsieur Jean AZEMA
Monsieur Mohamed BEN AHMED
Monsieur Jean-Pierre BRAQUELAIRE
Monsieur Bertrand DAVID
Monsieur Pierre FALZON
Monsieur Bernard PEROCHE

A mes parents
A tous ceux que j'aime...

Remerciements

Je voudrais remercier tout spécialement M. Bertrand DAVID, professeur à l'Ecole Centrale de Lyon et M. Jean-Pierre BRACQUELAIRE, professeur à l'université de Bordeaux I, d'avoir accepté, d'aussi bonne grâce, d'être rapporteurs de cette thèse. Qu'ils en soient chaleureusement remerciés ici, ainsi que pour leur aide précieuse pour l'amélioration de ce rapport.

Merci aussi à M. Jean AZEMA, Maître de conférence à l'université de Saint-Etienne, pour avoir accepté de faire partie de ce jury.

Je tiens à remercier M. Pierre FALZON, professeur au CNAM, pour sa présence dans le jury et pour sa gentillesse tout au long du déroulement de ce projet. J'apprécie sa présence. Merci.

Je tiens à témoigner ma gratitude à M. Mohammed BEN AHMED, professeur à l'Université de Tunis, qui m'a enseigné, m'a écouté et m'a fourni toute l'aide nécessaire au cours de mes études. Je suis profondement touché par sa présence dans ce jury. Pour tout cela, merci.

Merci à M. Bernard PEROCHE, pour avoir bien voulu m'accueillir dans son laboratoire, pour avoir accepté de diriger ces recherches, pour m'avoir entouré de ses conseils et avoir incité de nombreuses réflexions et apporté des solutions à ce projet. Je lui en suis reconnaissant.

Je n'oublie pas toutes les personnes du département informatique, qui pendant ces années m'ont procuré l'immense joie d'être parmi eux. Je pense particulièrement à Ehoud, Jori, Marie-Line, Dominique, Jean-Michel, Benou, Bernard (j'espère qu'il sera sur pied bientôt), Michel, Jean-Pierre, Florence, Samy, Annie, Jean-François, J.J, Gabi, Roland, Christine, Laurent, Nahed, Antoine, Marc, Philippe, Pascale, Elisabeth, Jean-Paul et le autres. Merci à tous.

Et puis, il y a des mercis pour tous ceux qui me sont tellement proches, et qui ont eu à supporter mes caprices. Je pense tout particulièrement à mon frère et aussi à Martine, Elisabeth et Fethi. Un grand merci à tous.

TABLE DES MATIERES

Liste des figures

LA LISTE DES FIGURES

INTRODUCTION

INTRODUCTION

L'intérêt de la conception des interfaces homme-machine a été longtemps sous-estimé et l'application sous-jacente était considérée comme la partie la plus importante d'un logiciel. Depuis ces dernières années, la situation tend à s'inverser; on considère l'application et l'interface comme deux composantes importantes d'un logiciel dont il faut tenir compte dès la première étape du développement.

Il existe plusieurs raisons qui ont fait que l'on s'intéresse de plus en plus aux interfaces homme-machine. Premièrement, l'apparition sur le marché de matériels informatiques performants à des prix "relativement" bas comme les stations de travail, qui possèdent une capacité mémoire importante (quelques Méga-octets), une rapidité d'exécution de l'ordre de quelques "Mips", un écran graphique haute résolution,... Deuxièmement, l'avancée des recherches en informatique, tant au niveau logiciel que matériel, l'émergence des sciences ergonomiques et l'apparition d'une population d'utilisateurs potentiels plus importante et plus variée comprenant des clients exigeants ayant des connaissances différentes. Toutes ces raisons ont contribué à faire de la technologie des interfaces un domaine stratégique pour la recherche et pour l'industrie.

La conception et le développement d'une interface homme-machine sont des tâches difficiles et coûteuses. La difficulté provient de la nature même de l'interface qui requiert des compétences et des techniques multiples où interviennent l'informatique, l'ergonomie et la psychologie cognitive. Ceci nécessite une architecture traduisant une organisation précise servant de support pour une méthode de développement; celle-ci doit être complétée par des outils rendant la méthode opératoire comme les boîtes à outils et les systèmes de gestion d'interfaces.

Au sens large, une interface homme-machine est un moyen permettant à un utilisateur de communiquer avec la machine. Initialement, chacune des deux entités (l'homme et l'ordinateur) s'exprime dans un langage spécifique: signal électrique, langage de programmation, langue naturelle, graphisme...Pour que la communication soit possible, il faut que le dispositif assure la traduction dans les deux sens.

Des interfaces utilisant soit le graphique, soit le langage naturel, soit un autre mode de communication ont déjà été développées pour offrir à l'utilisateur une convivialité permettant l'accomplissement d'une tâche avec le concours d'un ordinateur. Cependant, aucune des approches utilisant un seul mode de communication n'est totalement satisfaisante et l'idée de combiner plusieurs modes d'interactions est apparue vers les années 80. En effet, à l'exception d'un nombre limité d'utilisateurs pour qui l'efficacité d'un logiciel prime sur le mode d'interaction, la grande majorité des utilisateurs préfèrent un système convivial et facile à apprendre. Ces deux propriétés supposent la préservation des habitudes de communication de l'utilisateur, qui naturellement, combine différents modes.

Il est clair que la technologie des interfaces est un domaine de recherche intéressant qui offre l'opportunité d'innover en créant des interfaces intelligentes. Notons toutefois que les interfaces actuelles souffrent d'une lacune de taille qui est l'absence d'intégration des différents modes de communication disponibles pour l'utilisateur, et que la recherche dans ce domaine n'a pas beaucoup avancé.

L'un des objets de ce travail est la construction d'un outil graphique dans le cadre du projet MMI2[1]. MMI2 est un projet Esprit II qui a pour objectif principal de construire une interface homme-machine intelligente intégrant plusieurs modes de communication: langue naturelle (anglais, français, espagnol), langage de commande, graphique et gestuel. Ce travail est le résultat d'une approche expérimentale de développement d'un outil de manipulation directe, batisé *Outil de Dessin*, dans un environnement multi-mode. Dans ce contexte, notre démarche a consisté, d'une part, à utiliser des techniques de construction géométrique dans la conception d'une interface. Nous proposons pour cela un modèle fondé sur le concept de carte planaire. Ce concept, introduit par des mathématiciens [Edmo 60] et dévelopé par Cori [Cori 75], a rapidement trouvé des applications en informatique [Mich 84]. Il conduit à des structures de données de type polygonal dont la notion de base est le *brin*, qui est une ligne ou arête topologique orientée. Il nous permet dans un premier temps d'établir un algorithme de saisie et modification de cartes planaires qui, tout en contrôlant à chaque instant la planarité de la carte, est capable de détecter automatiquement la fermeture des polygones. D'autre part, nous nous sommes intéressé à l'étude de la sémantique graphique en tenant compte de l'intégration des différents modes de communication pour assurer un dialogue multi-mode entre l'utilisateur et l'ordinateur.

Le mémoire est organisé en quatre chapitres:

Le chapitre I présente le projet MMI2 et situe le travail effectué dans le cadre de ce projet.

Cette présentation donne un aperçu sur le système MMI2 et les différents modules qui le composent. Elle permettra aussi de situer l'outil graphique dans le système et de définir le moyen de communication entre les différents modules (appelé CMR).

Le chapitre II présente l'état de l'art des interfaces.

Ce chapitre donne une vision globale de la nature d'une interface homme-machine. Le problème de la conception d'interfaces réellement adaptées aux utilisateurs ne peut être résolu par l'informatique seule, ou la psychologie seule ou l'ergonomie seule. Il ne peut être abordé sérieusement qu'en intégrant des résultats et des connaissances de ces disciplines. Aussi, dans ce chapitre, différents modèles, formalismes et outils de développement sont présentés. Enfin, on parlera des interfaces multi-modes.

1. MMI2: A Multi-Mode Interface for Man-Machine Interaction with Knowledge based systems

Le chapitre III est consacré aux constructions géométriques et plus particulièrement aux cartes planaires.

Ce chapitre rappelle les définitions d'une carte planaire et présente un algorithme de construction incrémental d'une carte planaire. Il évoque les inconvénients de telles structures de données dans un environnement interactif qui sont dus essentiellement à l'imprécision numérique.

Le chapitre IV décrit la conception et la réalisation de l'outil de dessin.

Ce chapitre présente les structures de données adoptées et les algorithmes importants pour la construction de l'outil de dessin. La conception de cet outil est abordée selon des critères de flexibilité et de réutilisabilité. On précisera notamment la dépendance de l'outil vis-à-vis de l'application et vis à vis du système de fenêtrage.

La conclusion dresse un bilan des apports de ce travail. Il s'agit principalement d'une contribution à la conception d'outils graphiques dans un environnement multi-mode ainsi que l'utilisation des techniques de constructions géométriques pour la conception d'une interface interactive. La conclusion présente ensuite les extensions et les améliorations futures de l'outil de dessin.

CHAPITRE I

LE PROJET MMI2

Cette présentation donne un aperçu sur le système MMI2 et les différents modules qui le composent. Elle permettra aussi de situer l'outil graphique dans le système et de définir le moyen de communication entre les différents modules (appelé CMR).

I- LE PROJET MMI2

I-1. INTRODUCTION

MMI2 (Multi-Mode Interface for Man_Machine Interaction) est le projet Esprit II N° 2474, qui a débuté en janvier 1989 pour une durée de cinq ans. Les équipes qui participent à la réalisation de ce projet sont: BIM (Belgique) le leader du projet, L'Ecole des Mines de Saint-Etienne (EMSE), l'Institut National de Recherche en Informatique et en Automatique (INRIA), le CRISS (Grenoble-France), Rutherford Appleton Laboratory (Angleterre), l'Université de Leeds (Ecosse) et ISS (Espagne).

MMI2 a pour objectif principal de construire une interface homme-machine pour une base de connaissances qui intègre plusieurs modes de communication: des langues naturelles (anglais, français et espagnol), un langage de commande, le graphique et le gestuel. Le choix de ces modes est motivé par les besoins actuels: à l'exception d'un nombre limité d'utilisateurs qui donnent plus d'importance à l'efficacité d'un système qu'à la forme du dialogue, la grande majorité des utilisateurs préfèrent les systèmes conviviaux. Cette convivialité suppose la préservation des habitudes de communication de l'utilisateur, d'où la nécessité d'utiliser des modes naturels de communication que sont les langues naturelles, le graphique ou le gestuel. Cependant, par souci d'efficacité, un langage de commande est également prévu.

A travers un tel projet, les équipes participantes doivent travailler sur les points suivants:

- *La compréhension de la langue naturelle*: ceci implique le développement d'analyseurs et de générateurs des langues naturelles utilisées. L'un des objectifs du projet dans ce domaine est l'intégration, dans un seul système, des modules déjà existants pour la langue française et pour la langue anglaise qui utilisent des approches et des techniques différentes. Cette intégration sera obtenue par la définition d'une représentation sémantique commune à tous les modules du système. Un autre objectif du projet est le développement d'un module pour l'analyse et la génération de la langue espagnole.

- *La conception d'un langage de commande*: il doit être compatible avec les connaissances de l'utilisateur. Ces connaissances sont généralement de deux types: les connaissances de l'utilisateur concernant la langue naturelle, et celles concernant le domaine de l'application. L'existence de trois langues naturelles dans l'interface suppose que le langage de commande permette l'utilisation d'un vocabulaire extrait de chacun des différents langages existants dans MMI2 (par exemple, on doit pouvoir utiliser les commandes MOVE ou DEPLACE indifféremment).

- *Le graphique et le gestuel:* le but est de spécifier, concevoir et implémenter des outils graphiques et gestuels. Ces outils doivent, d'une part, être indépendants de l'application et d'autre part, tenir compte de relations avec la langue naturelle et le langage de commande (par exemple la manipulation d'objets graphiques à travers la langue naturelle ou le langage de commande).

- *La gestion du dialogue et le modèle utilisateur*: l'un des objectifs principaux de ce projet est de développer un gestionnaire de dialogue intelligent permettant un dialogue multi-modes. L'idée de base est que chaque entrée doit être traduite en une représentation sémantique adéquate. Le gestionnaire de dialogue doit décider de l'action à entreprendre, utiliser les stratégies de dialogue pour contrôler l'interface en faisant appel à un modèle utilisateur pour sélectionner le mode de réponse approprié.

- *L'intégration des modes*: la définition d'une représentation sémantique commune à tous les modes d'interaction est une tâche importante du projet. D'autres thèmes de recherche seront étudiés, tels que le passage progressif de la langue naturelle au langage de commande, l'intégration de la langue naturelle et du graphique permettant ainsi, non seulement la manipulation d'objets graphiques à l'aide de la langue naturelle, mais aussi l'utilisation de tableaux, d'histogrammes,...

- *La portabilité de l'interface*: l'interface devant être portable sur d'autres bases de connaissances en Prolog, une grande attention a été portée, lors de la définition de l'architecture, à la flexibilité en précisant les points d'interconnexion avec l'application et en développant des outils qui permettent une adaptation rapide de l'interface à une nouvelle application.

- *L'application*: le prototype de l'interface est connecté à une application spécifique. L'application développée au cours du projet, est un système expert pour la conception de réseaux informatiques, NEST[1]. Une telle application a une grande variété d'utilisateurs: techniciens, commerciaux, débutants, experts,... Dans cette application, on manipule des informations graphiques et textuelles. Cela justifie donc l'utilisation de la langue naturelle, du langage de commande, du graphique et du gestuel. Une caractéristique importante de l'application est qu'elle permet de tester tous les aspects de l'interface; en particulier tous les modes d'interaction choisis doivent être utilisables naturellement dans le contexte de l'application. Signalons également que nous devons déduire une méthodologie permettant d'adapter l'interface à de nouvelles applications. Un des objectifs du projet est de minimiser les transformations nécessaires dans ce cas aux parties strictement liées à la description sémantique de l'application.

1. Network Expert SysTem

Dans la section 2, nous présenterons l'architecture du système et nous décrirons ses différents modules ainsi que les structures de données les plus importantes qui leurs sont associées. Dans la section 3, nous parlerons du domaine de l'application qui est la conception de réseaux informatiques. Nous consacrerons la section 4 à la CMR (Common Meaning Representation) qui est le formalisme interne permettant la communication entre les différents modules du système.

I-2. L'ARCHITECTURE GENERALE DU SYSTEME

L'interface est constituée de plusieurs modules baptisés "experts" (voir la figure 1). Un expert est un module capable d'accomplir une tâche spécifique et qui possède ses propres structures de données. Tous les modules du système communiquent entre eux à travers un formalisme interne appelé CMR (Common Meaning Representation). On

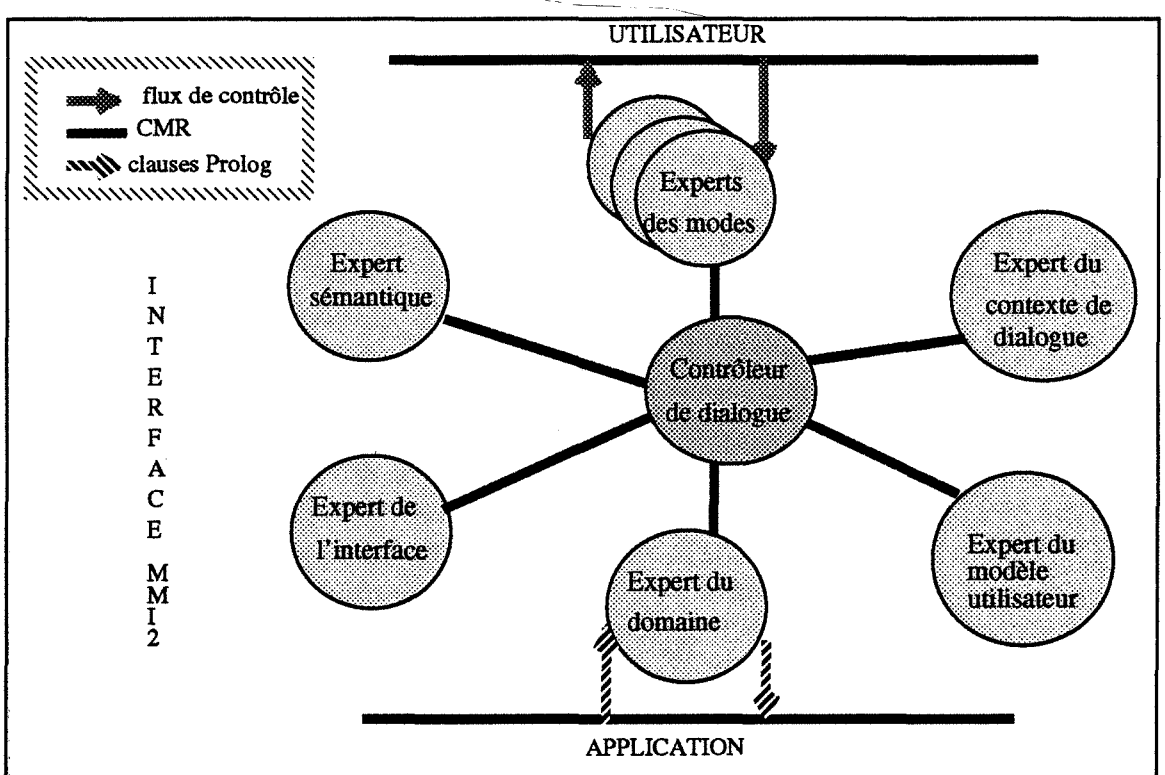


Figure 1: Architecture générale du système

remarquera que l'architecture du système est centralisée; en effet, la figure 1 illustre les interactions entre les différents experts et le contrôleur de dialogue. Ceci correspond au mode de fonctionnement du système où chaque requête doit passer par le contrôleur de dialogue pour être aiguillée vers l'expert approprié. Cependant, il est possible, selon les besoins, que les experts communiquent directement entre eux sans passer par le contrôleur de dialogue.

Nous allons présenter successivement le rôle et l'utilité de tous les modules, en fournissant quelques explications sur la manière dont ces modules ont été (ou vont être) implémentés.

I-2.1 Le contrôleur de dialogue: la flexibilité du dialogue

Sachant que l'application est un système expert qui a ses propres structures de données, nous pouvons dire qu'il y a un "*espace problème*" où les données initiales du problème à résoudre sont placées, un "*espace solution*" où le système expert peut construire sa solution et une base de connaissances contenant des connaissances générales sur le domaine. Que se passera-t-il quand l'utilisateur commencera à spécifier un problème? Comme notre application concerne les réseaux, supposons que l'utilisateur commence par décrire le nouveau réseau à soumettre à l'analyse de l'application. Doit-on alors envoyer à l'application chaque nouvel objet et le représenter dans "l'*espace problème*" ou bien doit-on le représenter au niveau de l'interface? La réponse à ces question n'est pas facile et les points de vue divergent. Tous ce que nous pouvons dire, c'est que ces questions constituent un axe de recherche important dans le domaine des interfaces homme-machine.

Nous pensons, pour des raisons de flexibilité, qu'on doit avoir une représentation du problème au niveau de l'interface. En effet, pour un système expert, la spécification du problème nécessite un ensemble d'informations de différents types. Si le dialogue est dirigé par l'application (application-driven), ces informations seront probablement demandées à l'utilisateur.

En réalité, dans un dialogue d'acquisition de données, l'utilisateur change de thème, répond à une question par une autre question, donne des réponses ambiguës et incomplètes qui nécessitent un sous-dialogue, demande une aide,... C'est le travail de l'interface et non de l'application de résoudre de tels problèmes. Si les données de l'utilisateur sont envoyées directement à l'application, elles pourraient être incomplètes ou momentanément erronées.

Ce type de problème a intéressé les chercheurs et peut être illustré par l'exemple suivant:

SYSTEME> *Combien voulez vous investir dans l'installation du réseau?*
UTILISATEUR> *parlons d'abord des machines Sun.*

En face d'une telle réponse, l'interface pourrait soit imposer un dialogue strict, soit autoriser le changement de sujet en supposant qu'elle puisse détecter un tel changement et revenir plus tard au premier sujet s'il est essentiel dans la formulation du problème.

Des problèmes similaires peuvent se poser avec une interaction graphique. En effet, l'utilisateur peut passer du temps à dessiner graphiquement un réseau, le modifier, manipuler des objets avant d'être complètement satisfait et de vouloir le soumettre à l'application. Si l'application devait analyser le dessin à chaque étape, avant que l'utilisateur pense avoir fini, cela poserait des problèmes. Toutes les interactions entre l'utilisateur et l'application, avant d'atteindre une formulation stable du problème, doivent être situées entre l'interface et l'utilisateur.

Pour permettre un dialogue du style de celui décrit ci-dessus, l'interface doit disposer de quelques structures de données: un "*plan de tâche*", un "*plan de communication*" et un "e*space de référence de discours*". La localisation de ces structures dans l'architecture est la suivante (voir la figure 2):

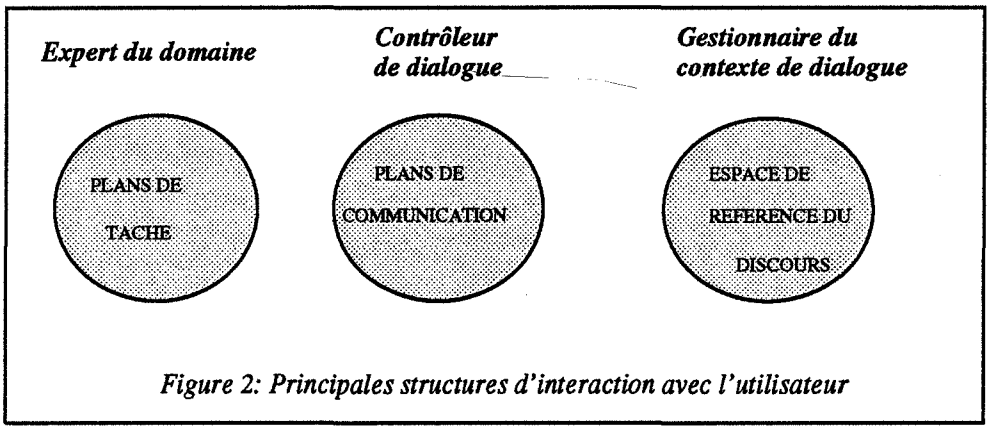


Figure 2: Principales structures d'interaction avec l'utilisateur

Le ***plan de tâche*** est un ensemble d'informations nécessaires à l'application pour spécifier un problème bien formé. Il peut être vu comme le squelette d'un problème typique qui sera instancié au cours du dialogue. Les *plans de tâche* sont bien sûr dépendants de l'application, et sont gérés par l'expert du domaine. Il existe plusieurs plans de tâche pour un seul domaine, correspondants aux différents types de problèmes qui peuvent être soumis à l'application. Normalement, les *plans de tâche* doivent être identifiés à travers l'interaction avec l'utilisateur.

Le ***plan de communication*** est une structure permettant de guider le dialogue avec l'utilisateur. Une fois la tâche identifiée, un *plan de communication* est activé pour demander les informations nécessaires pour le plan de tâche. Il existe au minimum un *plan de communication* pour chaque *plan de tâche*. Il peut y avoir d'autres *plans de communication* pour d'autres parties ou sous-parties du dialogue. Par exemple, il y a un *plan de communication* pour accueillir l'utilisateur et s'informer du problème qu'il veut résoudre. Comme les *plans de communication* contrôlent les structures de dialogue, ils seront sauvegardés par le contrôleur de dialogue dans une *librairie des plans de communication*. A tout moment, il y aura un *plan de communication* activé et il pourra faire les mises à jour du contrôleur du dialogue.

L'**espace de *référence du discours*** (focus space) décrit le contexte du dialogue, en particulier le contexte graphique. Cet espace contient non seulement les objets référenciés dans le discours, mais aussi les relations qui existent entre eux. Supposons, par exemple, que dans un contexte graphique, il existe un réseau avec plusieurs serveurs et plusieurs stations de travail; l'utilisateur peut agrandir (zoom) une partie du réseau qui contient un seul serveur. Ce serveur fait alors partie du contexte graphique courant.

D'après les remarques précédentes, le contrôleur de dialogue est le noyau du système MMI2. Parmi les structures propres à ce contrôleur, on trouve:

- Les **traces de contrôle** qui sont toutes les données relatives à l'état de l'interaction courante.
- Les **plans de communication** qui sont des structures permettant de guider le dialogue avec l'utilisateur.
- Les **attentes du système** qui concernent le déroulement futur du dialogue. Elles peuvent être générées à partir du plan de communication courant (voir la figure 3).

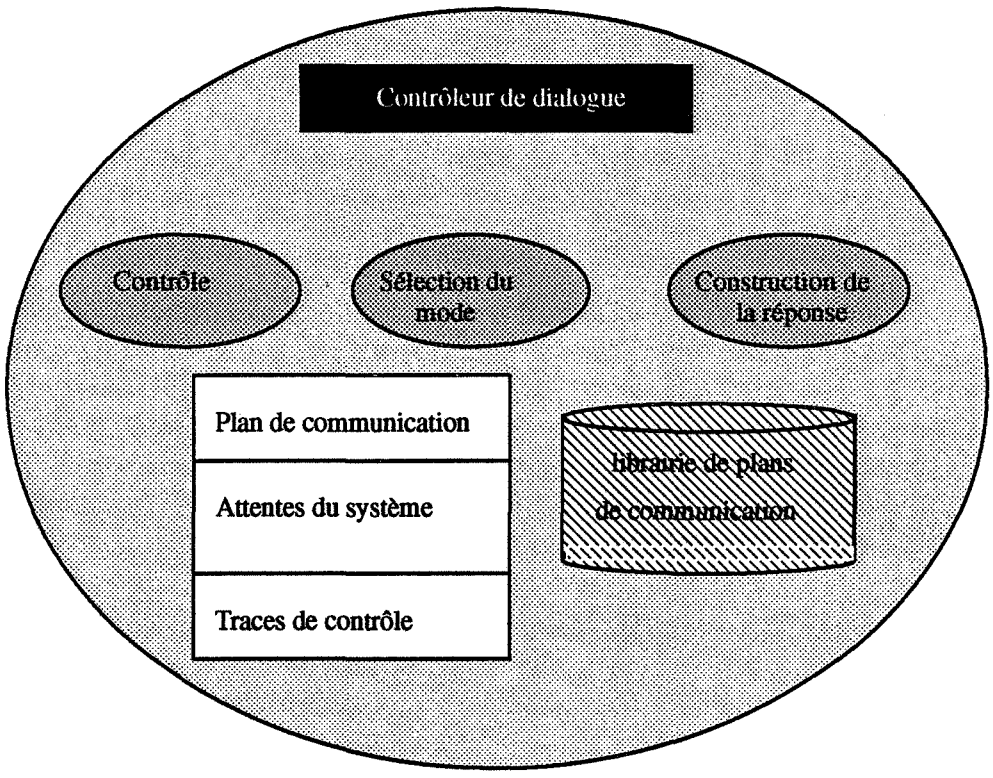


Figure 3: Architecture du contrôleur de dialogue

Dans le contrôleur de dialogue, il existe des processus qui gèrent ces structures de données et à tout moment décident de la stratégie à adopter. Le contrôleur de dialogue contient un processus appelé "*constructeur de réponses*" qui décide du contenu de la réponse

fournie à l'utilisateur. Il contient aussi un autre processus *"sélection du mode"* qui choisit le mode d'expression pour formuler la réponse. Évidemment, ces deux processus dépendent des structures de données du contrôleur de dialogue et du modèle utilisateur.

I-2.2 Le gestionnaire du contexte de dialogue: l'intégration des modes

L'architecture présentée ici est basée sur le fait que l'intégration des modes doit être gérée principalement par un expert assez général. Le gestionnaire du contexte de discours est basé sur la notion de *référence de discours* définie ci-dessous.

L'idée de base est que chaque interaction, ou chaque "discours" entre l'utilisateur et l'interface, quel que soit le mode utilisé, prend place dans *"un monde de discours "*, qui n'est pas nécessairement identique au monde réel, ou au monde de l'application. Chaque objet mentionné au cours de l'interaction doit exister, par le seul fait d'être mentionné, dans le *monde de discours* où il est appelé *"référence de discours"*. La *référence de discours* ne correspond pas nécessairement à un objet du monde réel: un utilisateur peut parler d'un serveur qui n'a pas encore été introduit dans le discours.

Plusieurs opérations (création, insertion, destruction...) sont appliquées aux *références de discours*. Cette notion est commune aux différents modes de communication. Voici des exemples concernant ces modes:

1) La langue naturelle

Les nouvelles références de discours sont introduites par des phrases nominales et référencées par des phrases verbales. Par exemple:

Ajouter un serveur au réseau. Connecter ce serveur à...

l'expression indéfinie "un serveur" conduit à l'introduction d'une nouvelle *référence de discours* dans l'*espace du contexte de discours*; de plus, cette référence doit avoir un identificateur unique. L'expression "ce serveur" peut alors référencer la nouvelle *référence de discours*.

2) Le graphique et le gestuel

Toutes les opérations graphiques ont un effet sur les *références de discours*. La création d'objets graphiques doit introduire de nouvelles références. La sélection d'un objet avec la souris doit manipuler une référence existante,... Voici un exemple d'intégration des modes:

<affichage graphique du réseau>
quel est le serveur?
< surbrillance de l'icône représentant le serveur>

La première action graphique installe un ensemble de *références de discours* dans l'espace de références de discours. La langue naturelle essaye de résoudre la phrase nominale "le serveur" en utilisant le *contexte de discours courant.* Le module de détermination des réponses décide du mode à utiliser (graphique) et de l'action graphique (surbrillance) appropriée pour répondre.

En résumé, l'idée de base pour l'intégration des modes est de permettre, au travers du gestionnaire du contexte de dialogue, de désigner un ensemble appelé *référence de discours* commun à tous les modes. Chaque mode possède ses propres mécanismes pour la création, la manipulation et la désignation des *références de discours.* Cependant, l'interface doit maintenir une représentation unique de la référence, même s'il y a plusieurs descriptions dans différents modes.

Le rôle de l'expert du contexte de dialogue consiste à identifier les structures du dialogue et ses différentes composantes, à enregistrer ces structures et à extraire les informations utiles des structures du dialogue. On a choisi une représentation des structures de dialogue par une arborescence de nœuds, où chaque nœud est une structure de données assez complexe.

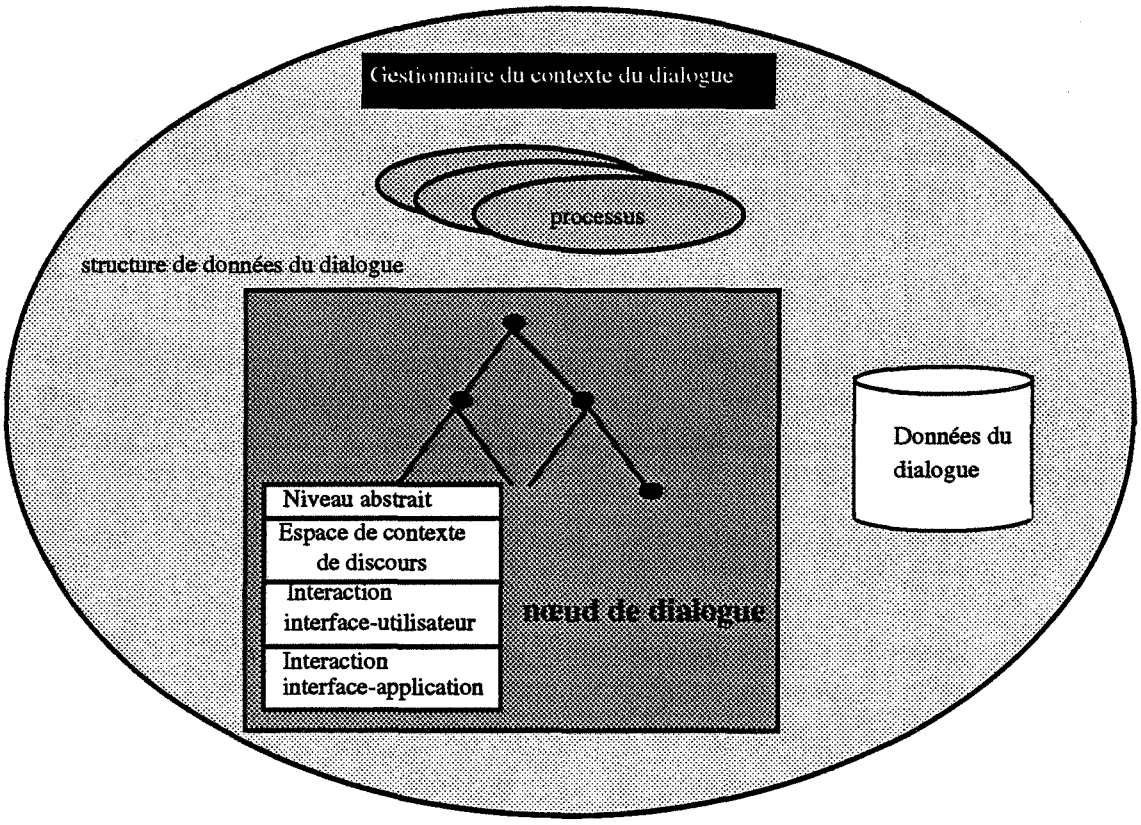
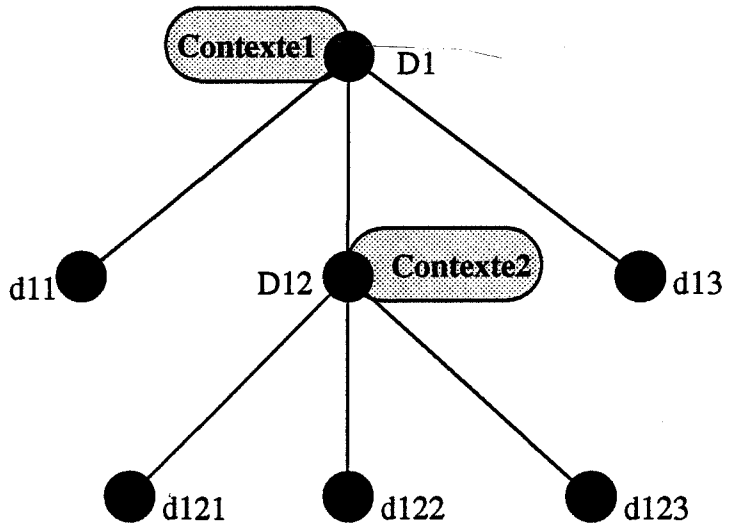


Figure 4: Architecture de l'expert du contexte de dialogue

La première zone d'un nœud de dialogue est un enregistrement détaillé des interactions utilisateur-interface et application-interface. La deuxième zone contient l'espace du contexte de discours courant (le contexte graphique inclus), et enfin une zone contenant des informations abstraites extraites de l'interaction.

Les structures de données doivent représenter tous les aspects du dialogue, des sous-dialogues, des démarches, des échanges,... Ces structures de données peuvent être vues comme un arbre n-aire où chaque nœud terminal représente une démarche ou un échange et les nœuds non terminaux un dialogue ou un sous-dialogue (d'autres types de nœuds peuvent être ajoutés). Seuls les nœuds de dialogue et les nœuds de sous-dialogue peuvent avoir un espace de contexte de discours. Les autres nœuds terminaux héritent du contexte de discours du nœud père le plus proche (voir la figure 5).



D1 est un dialogue.

D12 est sous dialogue.

d11,d13,d121,d122,d123 sont des nœuds terminaux.

le contexte de discours de ***d11*** est ***Contexte1***. Le sous-dialogue ***D12*** contient trois actions (***d121, d122, d123***) et un contexte de discours ***Contexte2***. Quand on quitte ***D12***, le contexte courant redevient ***Contexte1***.

Figure 5: Structures de données représentant les aspects du dialogue

I-2.3 L'expert sémantique: le type sémantique

La définition du type sémantique découle de deux remarques importantes:

1) La CMR ne doit ni être exprimée en langue naturelle, ni contenir des mots de la langue naturelle. La raison principale d'un tel choix est que la CMR doit être indépendante de la langue utilisée.

2) La CMR ne doit pas être exprimée directement en terme d'objets et de prédicats dépendants de l'application. A cela plusieurs raisons:

- Cela impliquerait que chaque mode doive assurer la traduction des données directement vers l'application. Ceci rendrait plus difficile la portabilité de l'interface, et poserait des problèmes de cohérence des données

- Le langage d'implémentation de l'application n'est pas nécessairement bien adapté aux besoins de l'interface. La hiérarchie des objets définis dans la réalisation de l'application n'est pas forcément utilisable par l'interface.

- Finalement, l'application ne contient pas tous les types nécessaires à l'interface. En effet, dans l'interface, on devrait pouvoir parler de l'interface elle même, des opérations sur l'interface,... De telles notions n'ont pas de correspondance dans le domaine de l'application.

En se basant sur ces remarques, l'interface doit posséder sa propre hiérarchie de types que nous appellerons *types sémantique*s de l'interface. Le type sémantique peut être lié à des objets (personne, ordinateur, serveur,...) mais aussi des actions (création, déplacement,...) ou des situations (être_connecté,...). Une partie de la hiérarchie est indépendante de l'application (les objets physiques, les actions); l'autre partie dépend du domaine de l'application.

L'introduction des types sémantiques implique les conséquences suivantes:

. Les types sémantiques et leur hiérarchie sont définis dans l'expert sémantique, qui est responsable de l'interprétation sémantique des entrées.

. Le contenu de la CMR est une notation logique utilisant les types sémantiques.

. La cohérence entre les types sémantiques et l'application est assurée par l'expert du domaine

. Chaque référence de discours doit avoir un type associé. Par exemple:

SUN4 (sun123)

où `sun123` est un identificateur de référence unique de type `SUN4`

La référence de discours d'une relation est dénotée par son type et ses arguments:

connected_to(SUN4(sun123),SUN3(sun124),ETHERNET_CABLE(cable125))

Une fois que nous avons défini les types sémantiques, nous pouvons les associer à des rôles, ou à des propriétés. De telles informations sont très utiles pour interpréter les entrées et détecter les incohérences et les formulations inadaptées sans avoir à accéder à l'application.

Dans l'exemple suivant, nous considérons l'action de connecter un ordinateur à un autre à travers un réseau. Le type de connexion doit être décrit comme suit:

connexion
origine: matériel
extrémité: matériel
instrument: câble

Supposons que la formulation de l'utilisateur soit la suivante:

connecter DALI à PICASSO avec un multiplexeur

Le système vérifie que DALI et PICASSO sont des noms d'ordinateurs dans le réseau, mais le multiplexeur n'étant pas un câble, l'action est refusée.

La discussion précédente implique l'existence d'un module séparé appelé Expert sémantique. Cet expert est concerné par la définition des types sémantiques, leurs propriétés et la sémantique des règles d'interprétation et d'évaluation des expressions.

I-2.4 - L'expert du modèle utilisateur

Les différences entre les utilisateurs peuvent influer sur le comportement du système qui doit s'adapter au mieux à leurs exigences. Pour s'adapter à ces différences, il faut modéliser les informations concernant les utilisateurs au lieu de compter sur un modèle unique implicite.

Le modèle utilisateur peut mettre en valeur le dialogue entre le système et l'utilisateur en influençant celui-ci de différentes façons:

1- par la façon d'exprimer les réponses du système: ceci peut être fait par la modélisation des préférences des utilisateurs.

2- par le contenu d'une réponse du système: par exemple, la réponse peut dépendre de ce que connaît (bien ou mal) ou de ce qui intéresse l'utilisateur.

3- par la compréhension de l'expression de l'utilisateur en connaissant son objectif et les informations dont il dispose.

4- par la façon de conduire le dialogue: en effet, le modèle utilisateur peut influencer la stratégie adoptée par le système pour contrôler le dialogue avec l'utilisateur. Par exemple, si l'utilisateur est un novice, le système peut prendre plus de contrôle que si l'utilisateur est un expert.

Cet expert contient donc des informations concernant les utilisateurs. Ces informations peuvent être utilisées par les différents modules du système. Le modèle choisi est un modèle orienté objets. A chaque classe d'utilisateurs est associée une description générale des connaissances qu'est censée posséder chaque membre de la classe. De plus, chaque utilisateur possède une description individuelle qui contient des informations qui lui sont propres, comme ses préférences, ses capacités, ses croyances. Des sous-modules appelés spécialistes permettent la mise à jour et l'extraction des informations utiles à partir de ces structures de données.

L'implémentation du modèle utilisateur est inspirée de GUMS (General User Modeling Shell) [Fini86] qui est basé sur une description individuelle de l'utilisateur et un héritage à partir de stéréotypes prédéfinis. La seule différence significative entre GUMS et le modèle utilisateur de MMI2 est que ce dernier permet un héritage multiple à partir de la hiérarchie des stéréotypes (voir la figure 6). Cependant, il existe une restriction sur l'héritage des stéréotypes qui ne doit pas contredire la description individuelle de l'utilisateur.

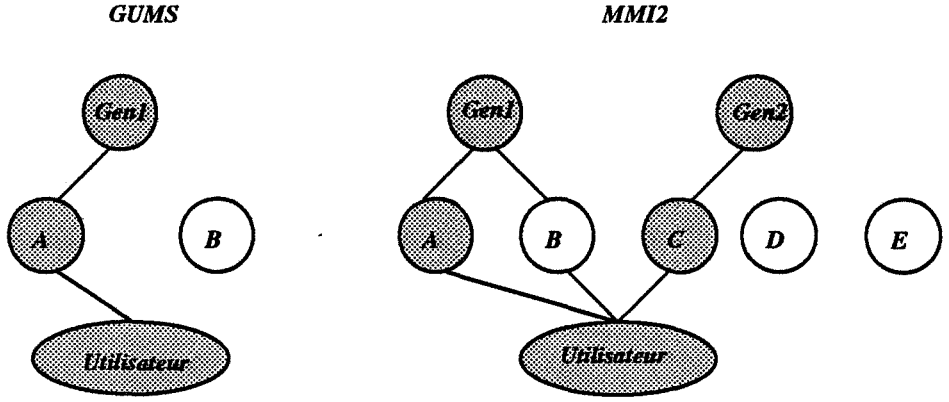


Figure 6: Différences entre GUMS et MMI2

L'architecture du modèle utilisateur est représentée sur la figure 7. La description des structures de données et des processus qui gèrent ces structures de données est fournie ci-dessous.

♦ *Les classes de description*: chaque utilisateur est défini par un ensemble de descriptions. Quelques unes de ces descriptions sont spécifiques à l'application. Par exemple, un utilisateur peut être expérimenté ou non sur la partie application du système. D'autres classes de description sont spécifiques à l'interface. Les informations concernant l'utilisateur, qui peuvent être extraites de ces descriptions, sont définies préalablement par le consortium sous forme de stéréotypes. Cependant, si un modèle utilisateur individuel est défini, alors, seules les informations concernant cet utilisateur qui ne coïncident pas avec le modèle du stéréotype seront maintenues. Il n'est donc pas nécessaire que le modèle individuel contienne ces informations générales qui seront de toute façon héritées du stéréotype.

♦ *Les préférences individuelles:* ces connaissances sont obtenues soit explicitement de la part de l'utilisateur, soit implicitement à partir du comportement de celui-ci. Ces connaissances doivent inclure des informations sur les options du dialogue (nom, sexe, statut, temps de réponse toléré par l'utilisateur,...), la langue naturelle préférée (anglais/français/espagnol, mots personnels à usage fréquent,...), les options du mode graphique (taille des fenêtres, préférence pour la représentation de données numériques par histogrammes ou par tableaux,...) et enfin les préférences concernant l'interface en général.

♦ *Objectif:* l'état final à atteindre qui sera déduit à partir des croyances de l'utilisateur, des plans de tâche de l'expert du domaine et des plans de communication du contrôleur de dialogue.

♦ *connaissances, présomptions et croyances:* on appelle connaissance une information que l'utilisateur possède sur le domaine et que le système connaît aussi. Par exemple, si l'utilisateur pense qu'il existe une imprimante sur le réseau et que le système le pense aussi, alors on peut dire que c'est une connaissance de l'utilisateur. Par contre si l'utilisateur pense qu'il existe un SUN3/60 sur le réseau alors que le système pense le contraire, ceci est appelé une croyance de l'utilisateur. Une présomption est une connaissance du système qui est nécessairement vraie. En résumé, supposons que U soit un utilisateur, A une proposition et S le système; intuitivement, on peut donner les définitions suivantes:

croyance: U pense que A et A est faux pour S
connaissance: U pense que A et A est vrai pour S
présomption: S pense que A et A et vrai pour U

♦ *le profil d'une classe d'utilisateurs*: chaque classe d'utilisateurs possède un profil dans lequel on trouve les connaissances et les présomptions concernant la classe.

♦ *le profil d'un utilisateur:* à la fin de chaque session; le modèle de chaque utilisateur est sauvegardé dans ce qui est appelé le profil de l'utilisateur.

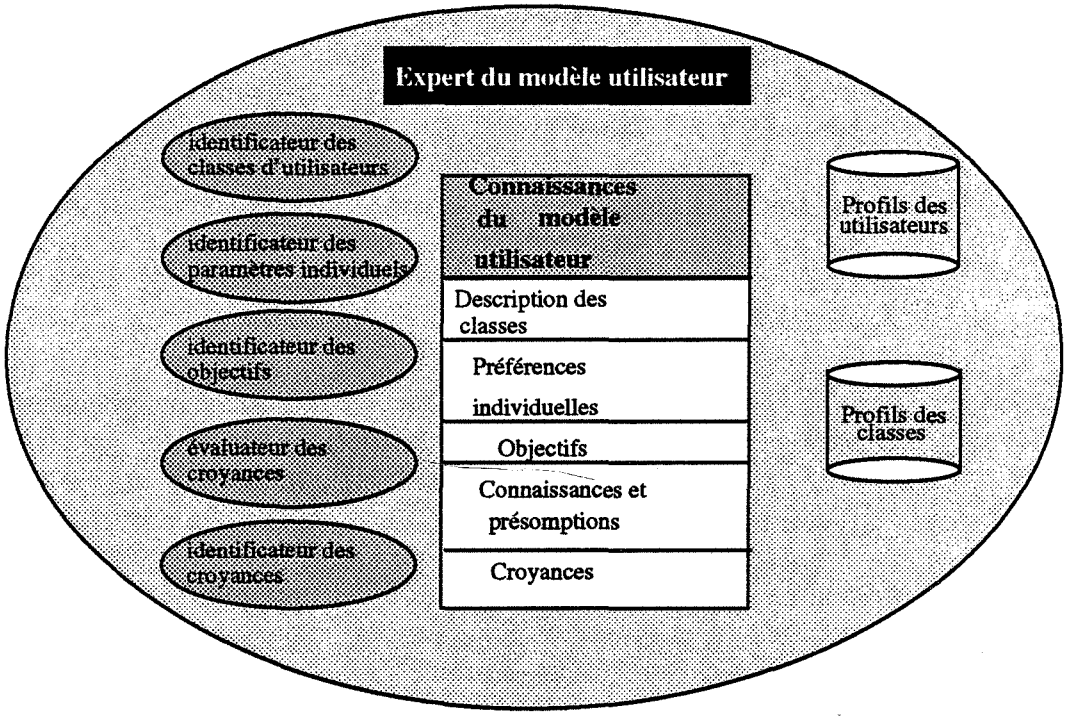


Figure 7: Architecture de l'expert du modèle utilisateur

Les processus contenus dans le modèle utilisateur sont:

♦ *l'identificateur des classes d'utilisateurs:* son rôle est de déterminer la classe de l'utilisateur en se basant sur les informations concernant l'utilisateur, ses connaissances, ses présomptions,...

♦ *l'identificateur des paramètres individuels:* il extrait les informations concernant l'utilisateur à partir du profil de la classe de l'utilisateur ou à partir des interactions entre l'utilisateur et le système, et envoie ces informations aux experts qui le désirent.

♦ *l'identificateur des objectifs*: il extrait l'objectif que l'utilisateur veut atteindre en se basant sur les croyances définies dans le modèle utilisateur, sur le plan de tâche de l'expert du domaine et sur le plan de communication du contrôleur du dialogue.

♦ *l'évaluateur de croyances*: son rôle est de comparer les croyances définies dans le modèle utilisateur avec les informations de l'expert du domaine. Il classe ainsi ces croyances en connaissances ou en présomptions.

♦ *l'identificateur de croyances:* à partir de la trace du dialogue et du plan de communication, il construit et insère les règles de croyances de l'utilisateur dans le modèle utilisateur. Ces règles de croyance dépendent aussi du profil de la classe de l'utilisateur.

La mise à jour des informations peut se faire de trois façons possibles:

1) un modèle prédéfini par le consortium: un modèle canonique est utilisé pour tous les utilisateurs et toutes les classes d'utilisateur.

2) le modèle est modifié explicitement: l'utilisateur est questionné sur ses préférences, ses connaissances...

3) le modèle est complété implicitement: le système construit le modèle à partir des interactions de l'utilisateur.

Enfin, le modèle utilisateur a des inconvénients qui sont:

- L'utilisateur peut introduire des informations qui n'intéressent pas le reste du système.
- Le temps nécessaire pour la construction d'une conclusion est souvent assez lent.
- La conclusion peut être erronée et conduit à un dialogue non "coopératif".

I-2.4.1 Acquisition des informations pour la construction du modèle utilisateur

Il est difficile de créer un modèle utilisateur en se basant uniquement sur l'intuition. Une situation de dialogue "homme-homme" est nécessaire pour analyser le comportement d'un utilisateur lors de la construction d'un réseau informatique. Une première expérience effectuée avec des experts humains s'est avérée peu satisfaisante: les experts se connaissent entre eux et, par conséquent, ils ont un modèle a priori de leur interlocuteur. De ce fait, une phase expérimentale a été mise en place pour établir le modèle utilisateur dans un environnement proche du réel. L'expérience en question a comporté deux étapes:

- simulation des interactions entre l'utilisateur et le système;
- enregistrement des remarques et des propositions des utilisateurs pendant l'interaction.

La méthode adoptée pour cette expérience ressemble à un magicien d'oz (Simulation d'un système par un expert humain). La seule différence entre cette méthode et le magicien d'oz est que les utilisateurs savent qu'en fait ils dialoguent avec un expert

humain et non avec un système expert. Les avantages de cette méthode sont les suivants:

- les interlocuteurs ne se connaissent pas: le dialogue est effectué via des stations de travail, et les interlocuteurs sont installés dans deux pièces différentes;

- l'interaction est analogue à ce que sera celle du futur système, puisqu'on utilise le graphique et la langue naturelle;

- tous les sujets ont à traiter les mêmes problèmes. Ces problèmes ont été définis pas le consortium, ce qui permet de fixer les facteurs "problème" et de comparer les protocoles de dialogue par rapport au niveau de compétence des utilisateurs.

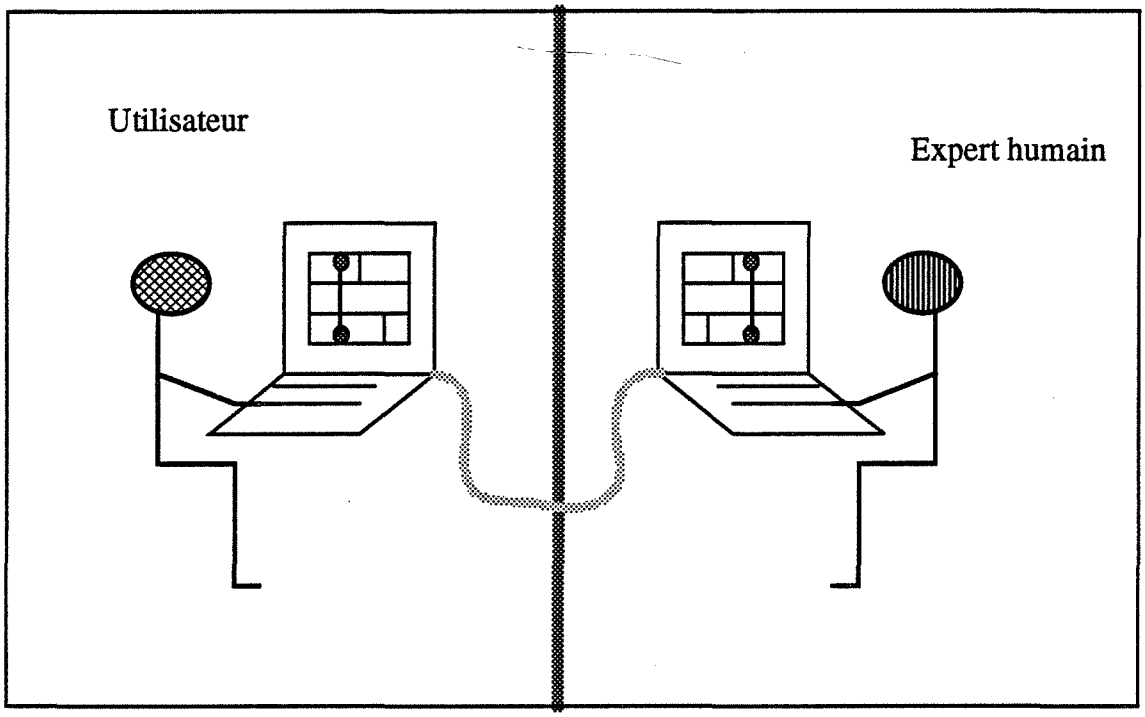


Figure 8: Expérience

Dans cette expérience, les utilisateurs communiquent avec un expert humain, à travers un clavier et un écran, afin de résoudre les problèmes qui leur sont soumis. On a choisi pour cette expérience plusieurs types d'utilisateurs (dix au total) qui ont un niveau de connaissance différent en matière d'installation de réseau et d'utilisation de l'informatique. Ces utilisateurs appartiennent à quatre catégories:

- Des technico-commerciaux spécialistes de la conception des réseaux au niveau physique (la topologie et les composants matériels du réseau).

- Des spécialistes des logiciels de réseau qui interviennent pour résoudre des problèmes de compatibilité entre les différents composants matériels du réseau.

- Des technico-commerciaux spécialistes de vente de matériels informatiques sans être pour autant spécialistes des réseaux.

- Des ingénieurs informaticiens qui ne sont pas des spécialistes des réseaux mais qui ont différents niveaux de connaissance dans ce domaine.

L'expert humain est un expert des réseaux qui a une très bonne connaissance dans le domaine de la conception des réseaux informatiques et occupe un poste de conseiller auprès des technico-commerciaux.

On a utilisé un logiciel appelé MUSK [Cram 87] (A MUltiuser SKetch and Talk Program) élaboré par RAL (Rutherford Appleton Laboratory) et adapté pour les besoins de l'expérience par l'Ecole des Mines de Saint-Etienne: création d'objets spécifiques aux réseaux (différents types de câbles, des boîtes,...) et sauvegarde automatique du dialogue. Ce logiciel est implémenté sur des stations Unix de type SUN connectées via Ethernet. Il permet de communiquer entre deux ou plusieurs stations du texte et du graphique en utilisant des routines Unix. En effet, tous les participants voient apparaître sur leur écran les mêmes objets; ainsi toute action effectuée sur une station apparaîtra instantanément sur les autres stations et réciproquement. On a donc placé un expert devant une station qui va jouer le rôle du système expert et on a pris une population d'utilisateurs auxquels on a proposé deux problèmes de câblage à résoudre en dialoguant avec l'expert via la deuxième station [Caho 89].

Le premier problème à résoudre consistait à concevoir un réseau pour un département de recherche et le second, à concevoir un réseau connectant plusieurs bâtiments d'une université. Chaque utilisateur a résolu successivement les deux problèmes. Toutes les dix minutes, on donnait à l'expert humain un questionnaire d'évaluation sur lequel il inscrivait des remarques concernant l'utilisateur, son niveau de compétence...

La deuxième phase de cette expérience a consisté à dépouiller le dialogue, chaque expert devant commenter la transcription de l'interaction (graphique et textuelle).

Les résultats de l'expérience ont été évalués à deux niveaux:

- Le premier niveau d'évaluation a consisté à localiser les connaissances de l'utilisateur en terme de connaissance correcte, incorrecte ou oubliée.

- Le deuxième niveau d'évaluation a consisté à classifier les utilisateurs selon leurs types de compétence.

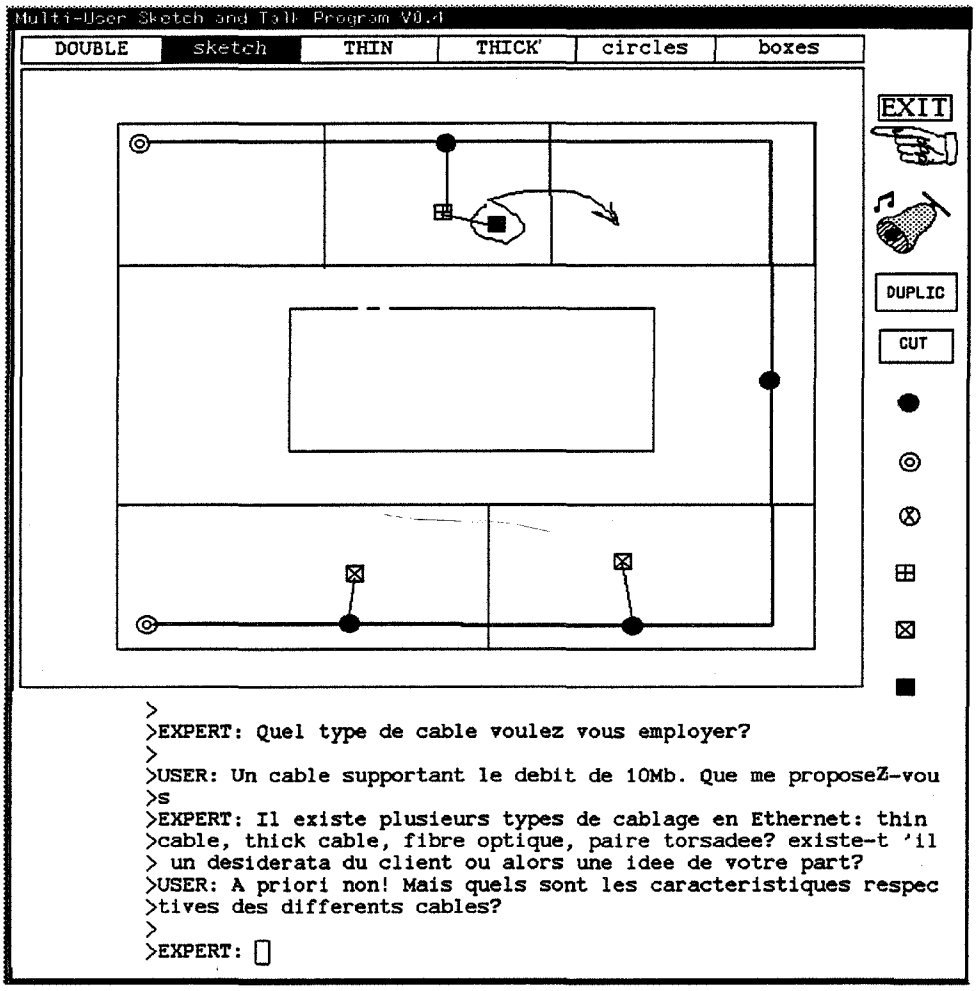


Figure 9 : MUSK

Cette expérience a permis au consortium de construire un modèle utilisateur dans un environnement proche de ce que sera le futur système. De plus, cela a permis d'extraire un sous-langage qui sera utilisé pour construire les analyseurs des langues naturelles car on ne peut pas actuellement construire un interpréteur efficace pour toute la langue naturelle.

I-2.5 l'expert du domaine: la connexion avec l'application

Cet expert possède toute la connaissance concernant l'application; on peut citer entre autres:

- les moyens de traduction d'une expression CMR (Common Meaning Representation) en clauses Prolog (cf section I-4);

- les moyens de création d'une planification des tâches décrivant la méthodologie à suivre pour résoudre le problème.

C'est l'expert du domaine qui assure la communication entre l'interface et l'application pour l'interrogation et la mise à jour de la base de connaissances. Essentiellement, il interprète les expressions CMR et les traduit dans la syntaxe du langage de la base de connaissances BIMprobe (c'est un langage logique, développé par BIM, sur lequel on a rajouté une couche objet).

Quand une expression CMR parvient à l'expert du domaine, un plan de tâche du domaine lui sera associé. La formule ainsi identifiée peut correspondre à une simple action sur la base de connaissances; elle sera alors exécutée directement. Dans le cas où la formule correspond à un plan de tâche complètement identifié, elle sera entièrement exécutée.

Dans le cas où le plan de tâche est identifié mais n'est pas complètement spécifié, un sous-plan de tâche est activé permettant ainsi de compléter le plan de tâche. Pour ce faire, l'expert du domaine doit activer un plan de communication approprié qui déterminera les informations que l'utilisateur doit compléter, ceci avec l'aide du modèle utilisateur. En effet, si l'utilisateur est un expert, une simple suggestion devrait suffire; si l'utilisateur est un novice, les questions seront plus détaillées.

Quand aucun plan de tâche n'a été identifié, l'utilisateur sera informé que le système ne comprend pas ce qu'il voulait faire et aucun accès à la base de connaissances n'est autorisé.

Pour bien illustrer le rôle de l'expert du domaine, prenons comme exemple la commande permettant de déplacer une machine d'une pièce à une autre (voir la figure 10).

1- Utilisateur: DEPLACE Machine0 Pièce3.

2- Le module du langage de commande vérifie la validité syntaxique de la commande.

3- Le module du langage de commande crée une expression CMR et vérifie la sémantique de la commande: il sélectionne dans son lexique le prédicat CMR qui représente la commande "DEPLACE". Il demande à l'expert sémantique si les arguments existent dans l'arbre sémantique. Après quoi, il demande à l'expert du domaine (qui consultera la base de connaissances) le type des arguments (dans ce cas la variable Machine0 est de type SUN3/60 et Pièce3 est de type LOCAL). Il demande à l'expert sémantique si les types des arguments vérifient les restrictions de la commande. L'expert sémantique donne son accord et ainsi le module du langage de commande crée une expression CMR pour la commande DEPLACE.

4- L'expression CMR est envoyée à l'expert de l'interface qui la renvoie au contrôleur de dialogue. Le contrôleur de dialogue envoie une copie à l'expert du contexte de dialogue et à l'expert du modèle utilisateur, interprète l'expression

CMR et l'envoie à l'expert du domaine.

5- L'expert du domaine exécute le prédicat DEPLACE(Machine0,Pièce3) qui déplace, dans la base de connaissances, la Machine0 dans Pièce3.

6- L'expert de domaine demande au module graphique de mettre à jour l'affichage en déplaçant la machine (Machine0) dans la pièce (Pièce3).

7- L'expert du domaine informe le contrôleur de dialogue. Celui-ci informe l'utilisateur que la commande a été exécutée en créant une expression CMR qui est envoyée à un module de la langue naturelle (le français).

8- L'expression est passée au module de la langue naturelle qui la transforme en français.

9- Le texte est alors envoyé à l'expert de l'interface qui l'affiche sur l'écran.

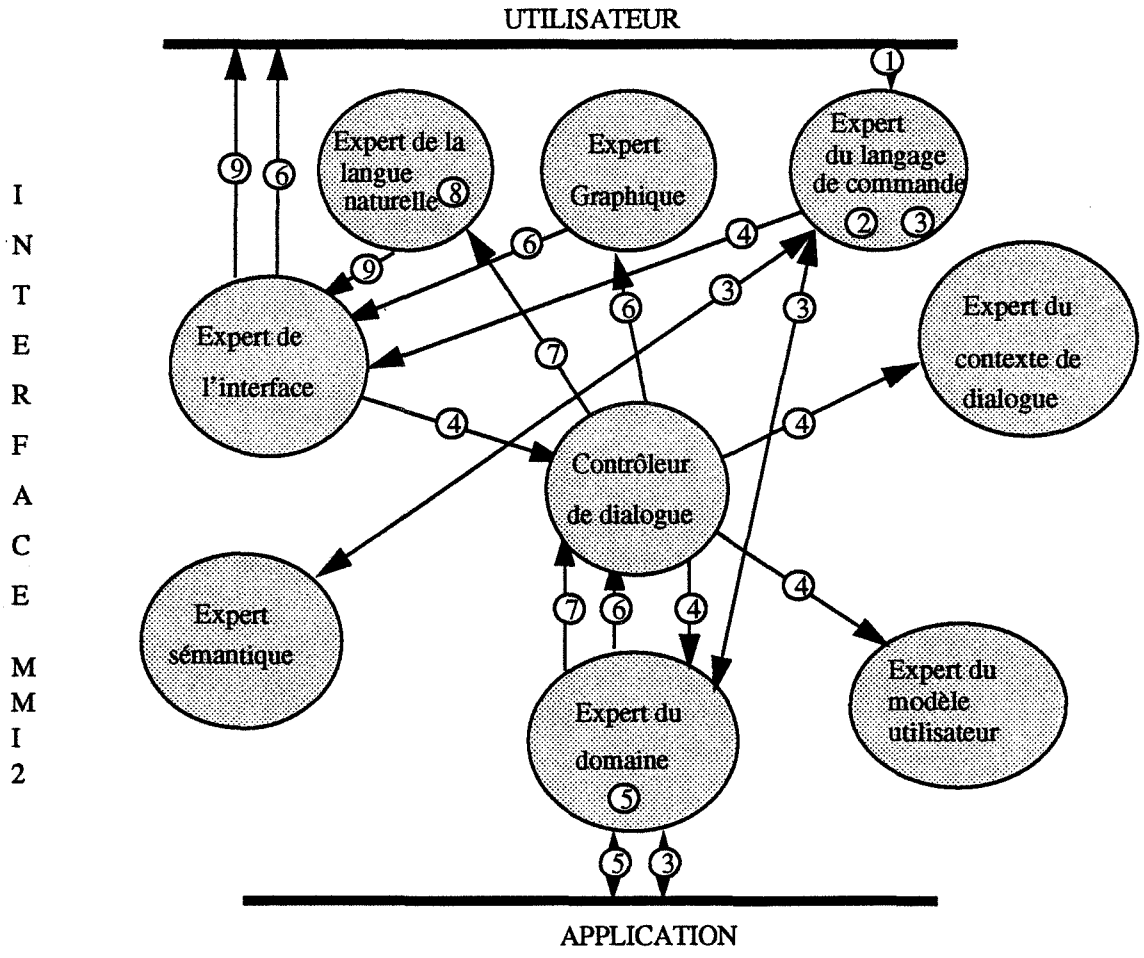


Figure 10: Les différentes phases d'une interaction

I-2.6 L'expert de l'interface

Dans MMI2, les informations déclaratives concernant l'interface font partie des connaissances de l'expert de l'interface. Ces informations sont consultées par le contrôleur de dialogue chaque fois que l'utilisateur demande des précisions sur les limites, les capacités, les structures et les composants de l'interface. Par exemple, si l'utilisateur pose la question: "quelles sont les langues naturelles disponibles dans l'interface?", l'expert de l'interface doit contenir la connaissance (anglais, français, espagnol).

La gestion de l'écran est aussi une tâche qui concerne l'expert de l'interface. En effet, dans le système MMI2, plusieurs fenêtres peuvent exister en même temps sur l'écran, d'où la nécessité de gérer intelligemment les positions des fenêtres sur l'écran et d'améliorer ainsi la flexibilité du dialogue.

L'interaction en langue naturelle ou avec le langage de commande est effectuée sur deux fenêtres distinctes. Ces fenêtres sont gérées par l'expert de l'interface. Une fois le texte saisi, l'expert de l'interface décide vers quel module il doit le transférer. Pour la langue naturelle, il doit choisir entre les modules des langues anglaise, française ou espagnole.

I-2.7 Les différents experts de modes de communication

Ces experts permettent d'effectuer les entrées/sorties pour chaque mode (langue naturelle, langage de commande, graphique et gestuel). Dans le cadre de cette thèse, seul le module graphique a été étudié en détail. Cependant, nous présenterons brièvement tous les modes de communication du système MMI2.

I-2.7.1 Les langues naturelles

Le projet MMI2 inclut trois langues naturelles: l'anglais, le français et l'espagnol. Pour chacune de ces langues, une approche différente a été choisie, pour les raisons suivantes. Premièrement, chaque langue possède ses particularités lexicales et syntaxiques. Deuxièmement, l'un des objectifs du projet est de réutiliser des travaux effectués dans ce domaine par les équipes impliquées dans le projet. Troisièmement, le fait de construire un support commun (la CMR) permet l'intégration des différents modes de communication et en particulier l'intégration des trois langues naturelles utilisées dans le projet.

I-2.7.2 Le langage de commande

Le langage de commande contient une partie syntaxique et une partie sémantique. La partie syntaxique réside dans l'expert du langage de commande alors que la partie

sémantique est contenue dans l'expert sémantique. La communication entre la partie sémantique et la partie syntaxique est assurée par la CMR.

I-2.7.3 Le gestuel

Le langage gestuel a pour objectif d'interpréter les gestes de l'utilisateur lors de l'accomplissement d'une tâche; il doit faciliter le dialogue entre l'utilisateur et le système. Le langage gestuel peut être considéré comme une sorte d'extension du langage graphique; en particulier, il devrait utiliser une bonne partie des outils développés pour le graphique.

Le langage gestuel permet à l'utilisateur de dessiner à main levée des symboles qui seront interprétés comme des commandes. Dans une première étape, 90 symboles ont été définis, dont quelques uns sont présentés sur la figure 11. Dans une deuxième étape, des méthodes ont été développées pour l'ajout de nouveaux symboles.

La principale difficulté de ce module est l'algorithme de reconnaissance des symboles qui est basé sur un arbre de décision. Si l'algorithme de reconnaissance échoue, un menu contenant tous les symboles est affiché sur l'écran, permettant ainsi à l'utilisateur de choisir sans ambiguïté le symbole désiré.

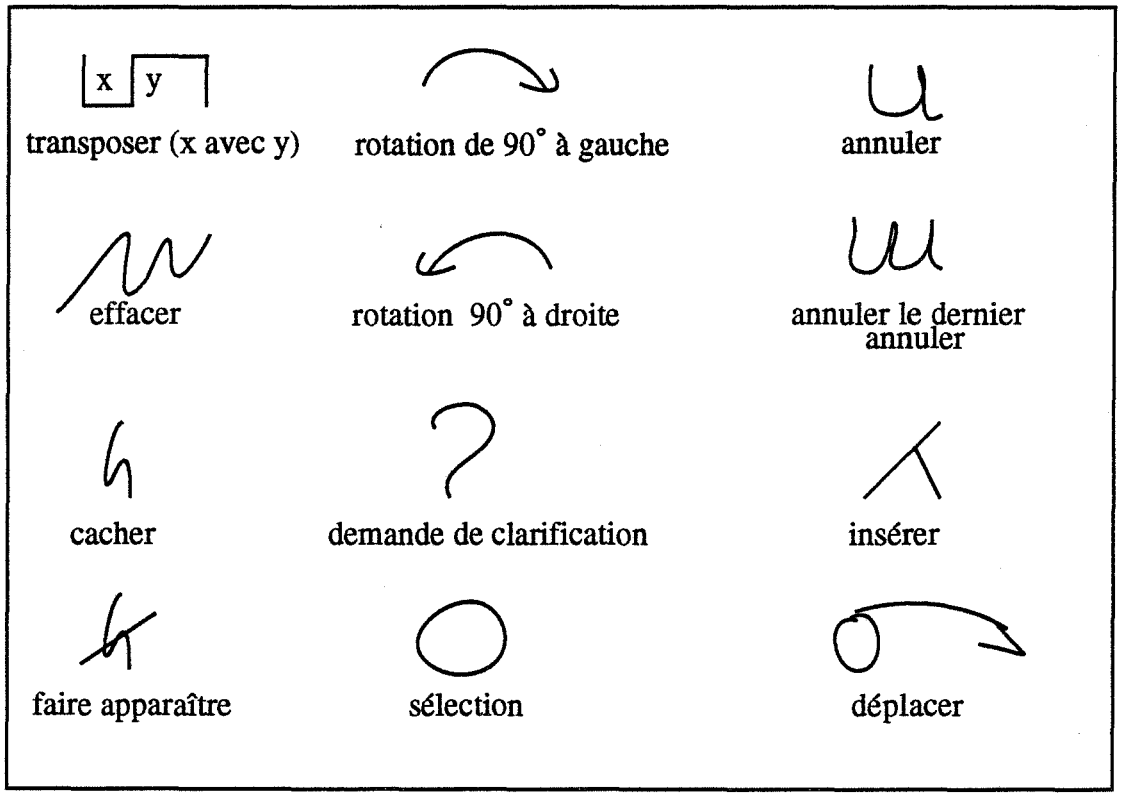


Figure 11: Quelques symboles du mode gestuel

I-2.7.4 Le graphique

D'après l'expérience effectuée avec le système MUSK, le client se présente devant l'expert en réseau avec des plans de bâtiment à câbler et des tableaux qui représentent des prix, des valeurs numériques, la charge du réseau à installer. Quelques-uns des plans des bâtiments sont détaillés; d'autres le sont moins (seuls les départements sont représentés). Durant la consultation, le client et l'expert discutent, dessinent, se réfèrent à des diagrammes.

L'analyse des résultats obtenus à l'aide de MUSK montre que le système MMI2 doit représenter des plans géométriques de bâtiments et des réseaux suivant plusieurs niveaux de détail. Les informations contenues dans les tableaux sont utilisées pour calculer des figures exactes, mais aussi pour montrer des prix, des longueurs, des capacités... Le système MMI2 doit donc être capable de présenter des tableaux, mais aussi des histogrammes, des camemberts, des graphes... Par exemple, les histogrammes doivent représenter des données numériques et permettre à l'utilisateur de les modifier. L'outil représentant le bâtiment et le réseau doit permettre la saisie du réseau et sa modification.

I-2.7.4.1 Architecture de l'expert graphique

Etant donnée l'architecture du système (voir la figure 1), l'expert graphique communique avec le contrôleur de dialogue au moyen d'un formalisme interne (la CMR). Ainsi l'expert graphique peut consulter des structures de données appartenant à d'autres experts.

L'expert graphique est capable d'interpréter (respectivement, de produire) la CMR qu'il reçoit (respectivement, qu'il envoie) au gestionnaire du dialogue. Par ailleurs, il fait appel au système de fenêtrage pour dialoguer avec l'utilisateur. Par conséquent, l'expert graphique est séparé en deux couches: l'une contient des outils graphiques de manipulation directe pour l'affichage et la modification des informations; l'autre contient les processus nécessaires pour transformer et produire la CMR. Les outils graphiques de manipulation directe doivent être d'utilité générale, et non limités à un domaine bien spécifique (la conception des réseaux informatiques en l'occurrence). Cette architecture doit permettre ainsi d'ajouter d'autres outils graphiques à l'interface sans aucune modification du gestionnaire graphique.

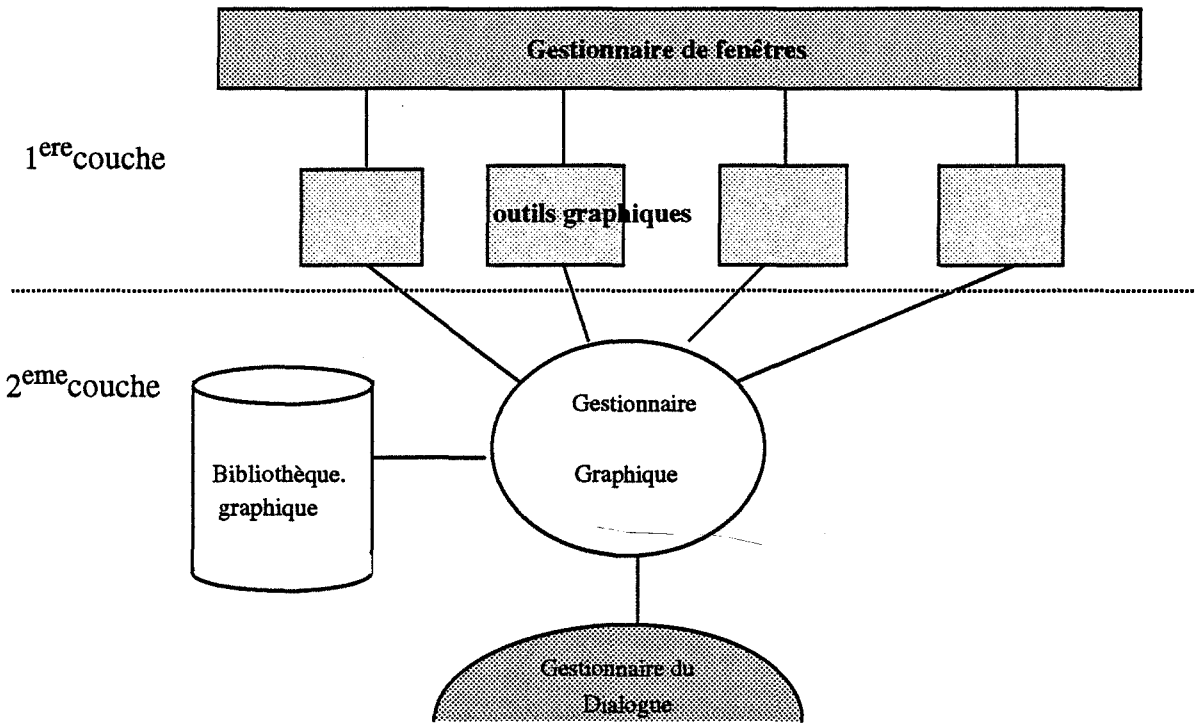


Figure 12: L'architecture de l'expert graphique

Du fait de l'application choisie, les outils qui sont à développer pour le projet doivent permettre deux types de représentation: une représentation topologique, exprimant les notions de connexité, de proximité, d'adjacence des objets,... et une représentation géométrique indiquant la forme d'un bâtiment, la longueur des câbles, les angles de courbure des câbles, l'emplacement des câbles dans les gaines et les structures du bâtiment, l'emplacement exact des machines dans les salles,.... De plus, des outils de représentation graphique de l'information doivent être développés. Ces outils doivent permettre la représentation d'histogrammes, de tables, de "camemberts" et de graphes que l'utilisateur peut manipuler interactivement. En conclusion, dans le cadre de ce projet, nous avons développé des outils pour:

- représenter la topologie des bâtiments et des réseaux;
- représenter la géométrie des bâtiments et des réseaux;
- représenter des informations (tables, histogrammes,...);
- manipuler des graphes.

Chacun de ces outils devrait permettre l'affichage de commandes venant du gestionnaire graphique et la saisie des données via la manipulation directe.

En général, l'expert humain dispose des plans des locaux fournis par le client. Deux alternatives étaient envisageables: la première consistait à utiliser un scanner pour saisir les plans des locaux, mais cette solution s'est avérée coûteuse et inutile car l'expert extrait des

plans seulement quelques informations pertinentes et fait abstraction des détails. La deuxième solution consistait à saisir à main levée sur les plans avec un stylet les informations qui sont utiles. Après cette saisie, l'utilisateur (l'expert) pourra faire des modifications ou ajouter d'autres informations en utilisant une bibliothèque graphique.

I-2.7.4.2 Le gestionnaire graphique

Le gestionnaire graphique communique avec les différents modules de l'interface de MMI2 à travers la CMR. La CMR doit donc être capable de représenter les actions produites par les outils graphiques et traduites par le gestionnaire graphique.

Pour illustrer le rôle du gestionnaire graphique, prenons un exemple où l'utilisateur veut ajouter une station au réseau en une position donnée. Dans ce cas, l'utilisateur doit sélectionner l'objet "station" dans une bibliothèque d'icônes et le positionner sur le réseau. L'outil graphique envoie alors un message au gestionnaire graphique que celui-ci traduit en CMR et envoie aux modules concernés du système.

Il reste néanmoins des problèmes à résoudre au niveau du graphique; en particulier, la relation entre le graphique et le langage naturel, l'utilisation de la CMR dans le graphique, et la consistance entre le graphique et l'application représentent des sujets de recherche où les solutions ne sont pas évidentes a priori.

I-2.7.4.3 Relations entre l'application et le graphique

Rappelons que l'un des objectifs principaux du projet MMI2 est la portabilité de l'interface vis à vis de l'application. De ce fait, le module graphique est complètement indépendant de l'application. Ainsi, le module graphique communique soit avec le contrôleur de dialogue, soit avec l'expert de l'interface en utilisant la CMR comme support de dialogue. Cependant, quelques opérations graphiques sont illégales vis à vis du domaine. Considérons, par exemple, l'opération de connexion de deux objets graphiques, où l'utilisateur sélectionne d'abord les deux objets à connecter puis choisit dans un menu l'item connexion. Si les objets choisis pour être connectés ne peuvent pas l'être physiquement, l'interface doit pouvoir arrêter l'opération et si possible proposer une réponse coopérative. Ceci est possible avec l'expert sémantique. En effet, l'opération de connexion est associée à un type sémantique "*connection*" qui vérifie des propriétés définissant une connexion "légale". Les objets graphiques sont eux aussi associés à des types sémantiques. Il est donc simple pour l'expert sémantique de détecter de telles inconsistances.

I-2.7.4.4 Relations entre la langue naturelle et le graphique

Dès la première étape de la conception, nous avons tenu compte des relations pouvant exister entre le graphique et la langue naturelle. Voici quelques exemples

d'interactions où apparait un mélange entre langue naturelle et actions graphiques:

1- le déixis apparent

Supposons que l'utilisateur saisisse la phrase suivante:

UTILISATEUR> *ajouter une station sun [click] ici.*

Sur cet exemple, le module graphique doit fournir la position topologique et géométrique du point désigné au gestionnaire du dialogue pour que ce dernier puisse mettre à jour les références de discours qui se trouvent dans l'expert du contexte de dialogue et permettre ainsi de résoudre le déixis.

2- le déixis caché

Ce problème se pose, par exemple, pour un réseau contenant plusieurs serveurs, dont un seul est apparent sur l'écran. L'utilisateur saisit la phrase suivante:

UTILSATEUR> *enlever le serveur.*

Pour résoudre cette ambiguïté, l'expert du contexte de dialogue maintient à jour une structure de données lui permettant de connaître la partie du réseau qui est affichée sur l'écran et de déterminer ainsi le "serveur" en question.

3-Les références spatiales

Supposons que l'utilisateur tape la phrase suivante:

UTILISATEUR> *ajouter une station à gauche de*
l'imprimante laser.

Ce cas doit être résolu par l'expert du contexte du dialogue en interrogeant l'expert graphique.

4- Référence à un objet par son aspect graphique

Le problème soulevé ici est la référence à un objet par ses propriétés graphiques. En effet, quand l'utilisateur fait référence indirectement à des objets en utilisant leurs descriptions graphiques, l'expert de la langue naturelle produit une requête sous forme d'une expression CMR et l'envoie au gestionnaire du dialogue. Celui-ci interroge l'expert sémantique qui l'informe qu'il s'agit d'un objet graphique. L'expert graphique est alors interrogé; il produit une expression CMR contenant la référence de l'objet.

Exemple:

UTILISATEUR> augmenter la barre la plus haute de 5%

L'expert du contexte de dialogue doit demander à l'expert graphique de transformer la description graphique en une description du domaine. Sur cet exemple, l'expert graphique doit interpréter cette phrase comme ceci:

"augmenter la capacité disque du serveur2345 de 5%"

I-3- LE DOMAINE DE L'APPLICATION

La conception des réseaux informatiques est un problème délicat, tant au niveau économique qu'au niveau technique. Le domaine des réseaux est très large et peut être divisé en deux parties: Le LAN (Local Area Network) et le WAN (Wide Area Network). La conception d'un LAN est assez différente de celle d'un WAN, le matériel et les règles de configuration utilisés n'étant pas les mêmes. En pratique, lors de la conception d'un WAN, le LAN ne sera pas étudié en détail; il sera considéré comme un noeud du WAN. Nous ne considérerons dans ce paragraphe que les réseaux de type LAN (le premier prototype du projet n'est concerné que par les LAN).

L'utilisateur doit dialoguer avec le système, en mixant les différents modes de communication, pour essayer de trouver une solution à un problème de câblage. Cependant, pour le même problème, il peut exister plusieurs solutions possibles. Le système doit être capable de guider l'utilisateur vers une ou plusieurs solutions, ou même proposer des solutions pour un problème donné. Les informations fournies par l'interface à l'application concernent:

. Les bâtiments
- aspect général et structures du bâtiment;
- nombre d'étages;
- dimension des salles;
- position des gaines techniques;
- emplacement des couloirs;
- ...

. Le réseau
- informations concernant un réseau préexistant (s'il y a lieu);
- type et position des machines et des câbles;
- position, fonction des serveurs;

De plus, le système peut avoir besoin d'informations telles que

- le type et le nombre de machines qui doivent être connectées;
- l'emplacement des prises de connexion;
- les applications qui doivent s'exécuter sur les serveurs;
- les performances escomptées (rapidité, temps de réponse,...);
...

Le système expert doit aussi tenir compte de plusieurs types de contraintes possibles. Les contraintes budgétaires que le concepteur ne peut dépasser, ce qui rend cette contrainte déterminante pour le choix de la solution finale: "La solution la plus chère n'est pas nécessairement la meilleure". Les contraintes techniques que le concepteur doit respecter comme les matériels disponibles, la compatibilité entre les machines, le protocole de communication entre les machines, la nature des câbles... Ces contraintes peuvent être modifiées suivant les progrès de la technologie informatique.

Ainsi le système expert de MMI2 contiendra les outils suivants:

un outil de conception:

Cet outil offre une aide pour la configuration de réseaux en proposant des solutions optimisées à un problème donné. Plus particulièrement, le système propose des solutions alternatives.

un outil interactif de mise à jour de la base de connaissances:

Les objectifs de cet outil sont doubles. D'une part, permettre une mise à jour facile de la base de connaissances, ce qui autorisera le système à évoluer avec la technologie; d'autre part, fournir au système une certaine capacité d'apprentissage. Ainsi, si des erreurs ont été observées pour un problème donné, on pourra modifier la base de connaissances afin d'éviter de refaire les mêmes erreurs.

un outil d'analyse:

L'outil interactif permet des modifications manuelles des solutions et l'outil d'analyse permet ainsi d'analyser les conséquences de telles modifications par rapport à différents paramètres tels que le coût, le trafic, le temps de réponse,...

un outil de sauvegarde:

Cet outil permet de sauvegarder les solutions trouvées, qui pourront être utilisées comme références pour résoudre de nouveaux problèmes.

un générateur de documents:

Les experts ont besoin de fournir à leurs clients une documentation complète sur la conception de leur réseau. Le générateur de documents permet de produire une documentation en se basant sur les solutions proposées et leur analyse. Cette documentation inclut une évaluation de la solution proposée ainsi qu'une table des composants, de leurs coûts et une estimation du coût global.

un outil didacticiel:

Enfin le système doit contenir un didacticiel permettant aux novices un apprentissage aisé de la construction des réseaux informatiques.

I-4 LA CMR

L'architecture du système MMI2 suppose que tous les experts (modules) travaillent en coopération pour, d'une part, comprendre les interactions de l'utilisateur et d'autre part, fournir des réponses, des questions, et des suggestions à celui-ci. Cette coopération nécessite un échange d'informations entre les différents expert du système d'où la nécessité d'avoir un formalisme interne commun à tous les modules pour véhiculer le contenu sémantique des interaction à l'intérieur de l'interface. De plus, ce formalisme appelé CMR (Common Meaning Representation), doit être utilisé comme un support pour le raisonnement sémantique et pragmatique.

L'objectif de la CMR est de fournir un langage permettant d'exprimer des actes de communication. Un acte de communication est la représentation de la sémantique d'une action de l'utilisateur ou du système. De plus, la CMR est aussi une structure de données que le système peut manipuler. En effet, la CMR permet d'identifier les objets, leurs propriétés et les relations qui existent entre eux quel que soit le mode utilisé. Il faut noter que la CMR comporte plusieurs aspects propres à la langue naturelle car celle-ci constitue la plus grande source de complexité sémantique.

Le choix effectué par le consortium du projet consiste à prendre comme base de la CMR la logique du premier ordre avec quelques extensions (le quantificateur "The" permettant d'utiliser des termes génériques, les quantificateurs numériques: exactement-3, moins-de-3, au-plus-3,...).

Avant de définir la CMR nous allons effectuer quelques remarques permettant de restreindre les concepts attachés à la CMR.

1) La CMR est une représentation intermédiaire, interne au système MMI2. Le rôle de ce formalisme interne est de contenir suffisamment d'informations sur le contenu sémantique des interactions.

2) La CMR n'est pas un formalisme se basant uniquement sur la logique du premier ordre. En effet, nous reconnaissons qu'il existe des aspects de la communication qui sont difficiles, voire même impossibles, à capturer par une approche purement logique.

3) Chaque expression CMR doit être interprétée par tous les modules du système. L'interprétation d'une telle expression est assez complexe et implique une gestion

du dialogue, un modèle utilisateur, une évaluation sémantique par rapport au modèle sémantique et par rapport au modèle du domaine.

Dans MMI2, un acte de communication peut provenir du mode graphique, du mode gestuel, d'une langue naturelle ou d'une commande. Cette action est effectuée soit par l'utilisateur soit par le système. La CMR est un langage permettant d'exprimer une action de communication.

Après une présentation générale de la CMR, la suite du paragraphe sera consacrée à l'utilisation de la CMR dans le module graphique.

I-4.1 Présentation de la CMR

Une expression CMR contient quatre types d'informations: un contenu propositionnel, une forme logique, une force élocutionnaire (C-force) et les annotations. Une expression CMR est aussi une structure de données qui peut inclure les informations additionnelles suivantes: état du processus, mode, temps de l'action, présuposition de l'utilisateur, erreurs de l'utilisateur, et d'autres informations syntaxiques.

Dans le tableau suivant, nous présentons quelques définitions qui nous serviront par la suite.

Symbole	Sens
V	*quantificateur universel*
E	*quantificateur existentiel*
n	*n est un entier, quantificateur de cardinalité*
THE	*quantificateur de description définie*
I	*opérateur iota de Russel*
->	*proposition conditionnelle (if-then)*
&	*conjonction*
v	*disjonction*
=	*identité*
F, G,...	*prédicat*
P, Q,...	*formule*
x:F	*variable x de type F*
x:F[P]	*F est une formule où x est libre, [P] est une restriction supplémentaire portant sur x*

I-4.1.1 La forme logique

Une expression CMR est une formule logique du premier ordre avec quelques extensions. Les extensions introduites sont: le quantificateur THE et les quantificateurs de cardinalité. Pourtant, il n'y a rien de prévu dans la CMR qui permette de représenter les noms pluriels; pour cela l'opérateur de composition (+) nous permet de former des noms de collection à partir de noms individuels et de définir une version plurielle d'opérateur singulier THE.

I-4.1.1.1 La logique typée

Nous désignons par logique typée une logique du premier ordre où les variables et les constantes peuvent appartenir à un certain type. Cette approche peut être vue comme une présentation syntaxique de la CMR pour que celle-ci soit la plus proche possible de la syntaxe de la langue naturelle. En voici quelques exemples:

(1) All men are mortal
(2) (Vx) (man(x)->mortal(x))
(3) (Vx:man) mortal(x)

Dans ces exemples, (3) est la version typée de (2). Il est aussi possible de typer une variable par une formule.

(4) All man that take poison are mortal
(5) (Vx:man[take_poison(x)]) mortal(x)

I-4.1.1.2 Le quantificateur THE

Nous avons introduit un identificateur distinctif appelé "THE" pour représenter un description définie. Par exemple:

(1) The leader of MMI2

cette phrase est représentée par:

(2) THEx: person[leads(x,MMI2)]

I-4.1.1.3 Les quantificateurs de cardinalité

Il existe cinq types de quantificateurs de cardinalité pour chaque entier. Par exemple, pour l'entier 3, les cinq types de cardinalité sont: exactement_3, moins_de_3,

plus_de_3, au_moins_3, au_plus_3.

exemple:
(1) Trois personnes travaillent sur le projet MMI2
(2) (3x:personne) (travail_sur(x,MMI2))

Soit P l'ensemble des personnes et W l'ensemble des personnes qui travaillent sur le projet MMI2. La phrase (1) vérifie la condition suivante:

$|P \cap W| = 3$

Le quantificateur de cardinalité est binaire. Malheureusement, il existe des quantificateurs de cardinalité qui font intervenir plusieurs ensembles d'individus:

trois personnalités accordent dix prix à douze champions

Cette phase met en relation trois ensembles d'individus: l'ensemble des personnalités, l'ensemble des prix et l'ensemble des champions.

Il est clair qu'une cardinalité ternaire pose des difficultés énormes pour l'analyse et l'interprétation de la langue naturelle. Ainsi, nous avons choisi de nous limiter à la cardinalité binaire pour la CMR.

I-4.1.2 Le contenu

En terme de logique, il existe une différence entre le contenu et la forme d'une formule. Le contenu est représenté par la partie non-logique de la formule, la forme est représentée par le reste de la formule [Sedl 90]. En général, pour exprimer les actes de communication, on en représente le maximum en terme de forme logique plutôt qu'en terme de contenu. Pour mieux comprendre le principe du contenu, prenons l'exemple suivant:

(1) Sebastian walked slowly.
therefore, Sebastian walked.

Une représentation logique de cette phrase est:

(2) walked_slowly(Sebastian)
therefore, walked(Sebastian)

cette formule n'est valide que si on a déjà introduit l'axiome suivant:

(3) (Vx) (walked_slowly(x)-> walked(x))

cependant une formulation différente (4), permet de capturer sa validité sans avoir besoin de (3):

(4) (Ex) (walking(x) & slow(x) & agent(x,Sebastian))
therefore, (Ex) (walking(x) & agent (x,Sebastian))

En effet, (4) nous a permis de capturer formellement la validité de (1). On dira aussi que (4) est la "réification" de (2).

I-4.1.3 les prédicats d'énonciation

Dans MMI2, le temps, l'aspect et la modalité sont traités et représentés par les prédicats d'énonciation. Les prédicats suivants ont été définis:

past(x), present(x), future(x)
state(x), event(x), process(x), habitual(x)
accomplished(x), unaccomplished(x)
culminative(x), nonculminative(x)

• *le temps*: le présent, le passé et le futur sont les trois prédicats liés au temps.

• *le type de situation*: quatre types de situation ont été définis:

* un état est une situation qui existe, est homogène, continue, durable et inchangée dans le temps.
* un événement est une situation instantanée, n'existant que pendant une très courte durée.
* un processus est une séquence d'événements ponctuée d'états; il est durable mais peut changer dans le temps.
* une habitude est un type de situation introduit pour lever certaines ambiguïtés de la langue française.

exemple:

```
John sings strangely
```

CMR_expr:

(Ex1: process[singing(x1)])
(Ex2: is_acc)
(Ex3: is_nonculm)

(Ex4: is_strange)
(present(x1) &
present(x2) &
present(x3) &
present(x4) &
agent(x1,John) &
subject(x2,x1) &
subject(x3,x1) &
subject(x4,x1)
)

• *l'accomplissement:* ce concept, essentiellement linguistique, est utilisé pour différencier une situation accomplie d'une situation qui ne l'est pas. En général, un événement est accompli par nature alors qu'un état ne l'est pas.

I-4.1.4 Les annotations

Pour pouvoir exprimer dans la CMR d'autres aspects qui ne peuvent apparaître dans le contenu ou dans la forme, on a défini une catégorie spéciale appelée "annotation", qui comprend des notions comme la chronologie, le mode de communication,....

I-4.1.5 La C_force

Le concept de C_force (communication_force) est défini en terme d'action et d'intention. La C_force est une combinaison de termes représentant l'intention d'un discours.

La grammaire de la C_force est la suivante (La grammaire complète est définie dans l'Annexe A):

c_force générale:

```
<gen_force)::= request| supply
```

c_force spécifique:

```
<spec_force>::= affirm | agree| confirm | deny |
     greeting| noncommit | nonact | nonclass |
hypothesis
```

sousforce:

```
<sub_force>::= action | inform | referent | explain |
     open |close | cut
```

les c_forces peuvent être combinées de la façon suivante:

```
c_force::= <gen_force> <spec_force>
c_force::= <gen_force> <spec_force><spec_force>
c_force::= <gen_force> <spec_force><sub_force>
c_force::= <gen_force><sub_force>
```

l'interprétation de ces notions est la suivante:

force générale: chaque interaction entre l'utilisateur et le système peut avoir deux types d'intentions possibles: l'une des deux parties fait une demande à l'autre et on l'appelle "request" ou l'une des deux parties veut fournir quelque chose à l'autre et on l'appelle "supply". Toute autre intention est exclue.

force spécifique: la différence entre une force spécifique et une force générale est qu'une force spécifique ne peut pas exister seule et qu'elle ne peut pas provoquer en tant que telle une action du système.

sous-force: Ces forces combinées à un autre type de force ont des répercussions directes sur le système, dans le sens où elles provoquent une action du système.

I-4.2 Utilisation de la CMR dans le module graphique

L'architecture de l'expert graphique contient un gestionnaire graphique qui gère la communication avec les autres experts du système MMI2 au moyen de la CMR. D'autre part, l'expert graphique contient des outils de manipulation qui communiquent directement avec le gestionnaire graphique par leur propre représentation. Le gestionnaire graphique transforme alors cette représentation en expressions CMR. La CMR doit être capable de représenter toutes les actions graphiques produites par ces outils. Voici deux exemples pour illustrer la traduction d'une action graphique en CMR.

Exemple 1

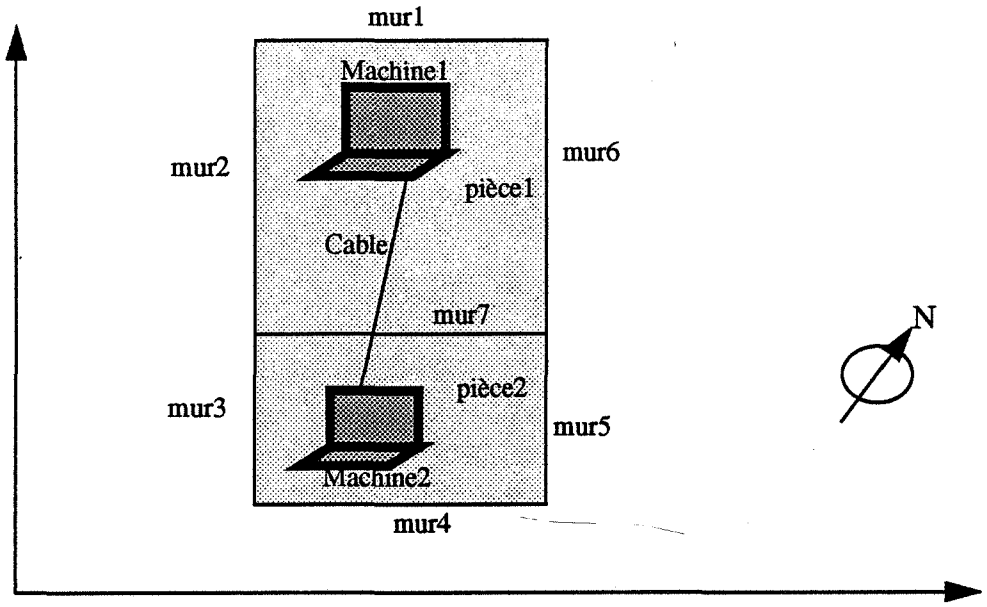


Figure 13: Exemple 1

Supposons que l'utilisateur ait saisi le dessin ci-dessus et qu'il veuille le soumettre à l'analyse de l'application. Le gestionnaire graphique doit produire l'expression CMR suivante:

On suppose que la pièce A est carrée et que la pièce B est rectangulaire

Acte de communication: insertion

```
/* pièce1 est carré de 7 mètres de côté */
Ex2:room [x1 = pièce1]
(
        is_square(x1)
&       Ex3: amount [x3=[meter,7]] wide(x1,x3)
&       Ex4: wall [x4 =mur1] interior_wall_of(x4,x1)
&       Ex5:wall [x5 =mur2] interior_wall_of(x5,x1)
&       Ex6:wall [x6 =mur6] interior_wall_of(x6,x1)
&       Ex7:wall [x7 =mur7] interior_wall_of(x7,x1)
)

/*pièce2 est rectangulaire de 7 mètres de largeur et 5 mètres de longueur */
Ex10 :room [x0 = pièce2]
(
```

```
        is_rectangular(x0)
&       Ex1: amount [x1=[meter,7]] wide(x0,x1)
&       Ex2: amount [x2=[meter,5]] length(x0,x2)
&       Ex3: wall [x3 =mur3] interior_wall_of(x3,x0)
&       Ex5:wall [x5 =mur4] interior_wall_of(x5,x0)
&       Ex6:wall [x6=mur5] interior_wall_of(x6,x0)
&       Ex7:wall [x7 =mur7] interior_wall_of(x7,x0)
)

/* pièce1 est à coté de pièce2*/
Ex1: room [x1 =pièce1]
Ex2:wall [x2= mur1]
Ex3:wall [x3= mur2]
Ex4:wall [x4= mur6]
Ex5:wall [x5= mur7]
Ex6: room [x6 =pièce2]
(
next_to(x1,x6) &
continuation_of (x3,x7) &
exterior_wall_of(x5,x8)
)

/*Machine1 est à 5 mètres de mur2 et à 4 mètres de mur7 */
Ex1: computer [x1=Machine1]
(
Ex2:wall [x3=mur2]
Ex3: amount [x3 = [metre , 5]]
        distance_from(x1, x2,x3)
&
Ex4:wall [x4=mur7]
Ex5: amount [x5 = [metre , 4]]
        distance_from(x1, x4,x5)
)

/* la machine2 est à 2 mètres de mur3 et à 4 mètres du mur7 */
Ex1: computer [x1=D]
(
Ex2:wall [x3=mur3]
Ex3: amount [x3 = [metre , 2]]
        distance_from(x1, x2,x3)
```

&
Ex4:wall [x4=mur7]
Ex5: amount [x5 = [metre , 4]]
distance_from(x1, x4,x5)
)

*/*Cable est le câble qui relieMachine1 et Machine2 */*
Ex1: ethernet_cable [x1=Cable]
(
Ex2: computer [x2=Machine1]
Ex3: computer [x3=Machine2]
link(x1,x2,x3)
)

Exemple 2

L'utilisateur veut ajouter une station à un réseau. Il sélectionne un icône représentant une machine et la place dans une pièce. Le gestionnaire graphique doit produire l'expression CMR suivante:

acte de communication: request
Ez: event[in(z)]
Ex: workstation
Ey: room[y=_12]
present(z) & present (x) &present (y) & accomplit(z) & objet(z,x) & localité(z,y)

Plusieurs systèmes multi-modes ont vu le jour ces dernières années. On peut citer, à titre d'exemple, le projet Esprit I ACCORD [Kri 89] qui intègre la langue naturelle et le graphique pour l'interrogation et la mise à jour d'une base de connaissances. Les différents modules du système communiquent entre eux à travers un langage de représentation sémantique commun InL. InL est une extension de la logique du premier ordre indexée. Le projet s'est heurté à un problème dû au fait que les objets graphiques sont traités comme des noms. Cette solution est appropriée pour la langue naturelle mais semble inadaptée pour le graphique, pour les raisons suivantes. Premièrement, dans le système graphique, les objets sont des entités et non des représentations graphiques des objets du domaine. Deuxièmement, InL tel qu'il a été défini ne permet pas d'exprimer les actions graphiques de la même façon que les actions en langue naturelle. Nous avons proposé une approche différente pour l'intégration de la langue naturelle et du graphique. Il peut sembler difficile a priori de représenter des objets graphiques complexes, comme un réseau informatique, en CMR. Notre approche ne consiste pas à représenter les objets graphiques en CMR, mais à décrire les interactions graphiques. Chaque action graphique ou geste de l'utilisateur (création, mouvement,...) est représentée dans le système par un acte de dialogue et une indication sur les objets concernés par l'opération. Par exemple, supposons que l'utilisateur

modifie la taille de la barre d'un histogramme représentant les performances d'une station de travail. L'expression CMR de l'action graphique doit décrire seulement l'action graphique; elle doit contenir les informations suivantes:

acte de communication: *demande (action(m))*
contenu propositionnel:
existe (m: changement_de_valeur)
patient (m, performance_de(station(Sun123), nombre(10))
valeur (m, nombre(20)).

Toutefois, dans MMI2, seuls les modes graphique et la langue naturelle sont utilisés en entrée et en sortie. Le problème qui se pose dans ce cas est le contrôle en sortie de ces modes par le contrôleur de dialogue. L'expert de la langue naturelle reçoit les expressions CMR du contrôleur du dialogue et les traduit en texte en langue naturelle. Cette traduction est basée sur l'interprétation des c-forces et des autres composants de la CMR pour produire un texte qui sera présenté à l'utilisateur. Ceci implique l'existence, dans l'expert de la langue naturelle, de règles qui permettent de générer des phrases à partir des expressions CMR.

Pour le mode graphique, le problème est plus complexe. D'une part, le graphique ne peut pas avoir le même statut linguistique en terme d'expressivité que la langue naturelle; d'autre part, il n'est pas capable d'interpréter tous les c-forces qui existent en actions graphiques. En effet, le système ne peut exprimer qu'un nombre limité d'actions en graphique.

Un autre problème est posé par le mixage des modes en sortie. En effet, supposons que le système affiche la question suivante:

"Quelle est la longueur de ceci?"

Le contrôleur de dialogue doit désigner l'objet en l'affichant en inverse vidéo, donc en envoyant l'expression CMR suivante à l'expert graphique:

acte de communication: *demande (action(m))*
contenu propositionnel:
existe (m: changement_de_statut)
patient (m, statut_de(mur (Mur23), inverse_vidéo).

L'inconvénient de cette approche est que cette expression CMR ne représente pas l'intention du système à l'égard de l'utilisateur (qui est dans ce cas de poser une question). Donc, le système peut utiliser un mode de communication pour converser avec l'utilisateur et utiliser un autre mode pour exprimer implicitement ses intentions, ce qui est en contradiction avec l'idée générale de MMI2.

I-5 CONCLUSION

Nous avons présenté dans ce chapitre le système MMI2 et les différents modules qui le composent. Il faut noter qu'un premier prototype du système complet a été intégré pour le mois de novembre 1991 et présenté lors de la semaine ESPRIT'91 à Bruxelles. Nous nous sommes efforcés pendant la réalisation de ce prototype de respecter les objectifs présentés au début de ce chapitre. Néanmoins, il reste des problèmes à résoudre tels que le passage progressif de la langue naturelle au langage de commande, l'enrichissement du vocabulaire des générateurs des langues naturelles et l'amélioration des performances des différents experts du système.

Notons toutefois, qu'il existe certains aspects qui ne sont pas pris en considération dans MMI2, notamment la datation des événements. En effet, dans MMI2, il a été prévu de dater les actions de l'utilisateur et du système en leur affectant la valeur de l'horloge au début de l'action. Pour la langue naturelle, l'unité événementielle est la phrase, ce qui pose des problème de synchronisation lorsqu'on utilise simultanément d'autres modes de communication. IPCdraw [Cael 91], qui est une interface multi-mode utilisant la langue naturelle, le graphique et la parole, apporte une solution à ce problème. Dans IPCdraw l'unité événementielle est le mot, complétée par le début et la fin de la séquence parlée. Pour mieux comprendre le principe de cette approche, prenons l'exemple utilisant la parole et la souris:

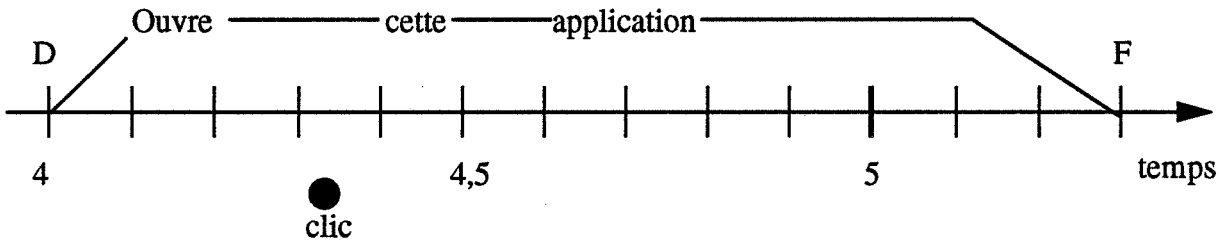


Figure 14: Datation des événements dans IPCdraw

Dans le cas de la figure 14 le déictique "cette" appelle une information de désignation complémentaire. IPCdraw recherche alors cette information parmi les autres événements multimodaux, dans un voisinage temporel (avant, pendant ou après) de l'événement "cette". Ici l'événement le plus proche est un clic de la souris qui fournit la référence à l'objet désigné.

CHAPITRE II

LES INTERFACES HOMME-MACHINE

Ce chapitre donne une vision globale de la nature d'une interface homme-machine. Le problème de la conception d'interfaces réellement adaptées aux utilisateurs ne peut être résolu par l'informatique seule, ou la psychologie seule ou l'ergonomie seule. Il ne peut être abordé sérieusement qu'en intégrant des résultats et des connaissances de ces disciplines. Aussi, dans ce chapitre, différents modèles, formalismes et outils de développement sont présentés. Enfin, on parlera des interfaces multi-modes.

II- LES INTERFACES HOMME-MACHINE

II-1 DEFINITION D'UNE INTERFACE

Généralement, on parle d'interface homme-machine (IHM) ou interface utilisateur pour désigner tous les aspects des systèmes informatiques qui incitent la participation de l'utilisateur à des tâches informatisées. Plus précisément, une interface homme-machine est la définition de toutes les entrées de l'utilisateur vers l'ordinateur, de toutes les sorties de l'ordinateur vers l'utilisateur et la définition d'enchaînements de telles entrées/sorties disponibles pour l'utilisateur.

Par opposition à d'autres machines qui représentent des extensions du corps humain, l'ordinateur, et en particulier le logiciel, représente plutôt une extension du cerveau humain, en faisant appel à des processus cognitifs, comme la perception, la mémorisation, le langage, la résolution de problèmes, la prise de décisions...

La qualité de l'interface est généralement déterminée par l'acceptation ou le refus du système par l'utilisateur. Pour des applications critiques comme le contrôle du trafic aérien, le contrôle des usines d'énergie nucléaire, la qualité de l'interface peut contribuer indirectement à des catastrophes.

Les systèmes graphiques sont presque toujours des systèmes interactifs. Les interfaces avec systèmes de fenêtrage, icônes et menus qui font usage du graphisme sont populaires, car elles sont faciles à utiliser, à apprendre et le temps d'apprentissage est relativement court. Ces interfaces graphiques existent dans plusieurs applications sur micro-ordinateur. Avec la chute des coûts du matériel et l'amélioration des capacités des ordinateurs personnels graphiques, nous sommes en mesure d'optimiser l'efficacité de l'utilisateur et celle de l'ordinateur.

Les produits actuels tendent à toucher une large population d'utilisateurs qui va du novice au spécialiste; cela a conduit les développeurs de logiciels à consacrer une part plus importante du logiciel et plus d'efforts à l'interface utilisateur. Cependant, plus l'interface est facile à utiliser, plus elle est compliquée à concevoir.

Une interface homme-machine s'étudie selon deux points de vue: celui de l'utilisateur du système informatique et celui du concepteur d'applications interactives. Il est important de séparer l'aspect utilisateur de l'aspect système où les objets et les primitives manipulés par le premier sont tout à fait différents de ceux manipulés par le second.

La qualité de l'interface dépend de ce que l'utilisateur voit de l'extérieur, donc de ce qu'il devrait connaître pour comprendre ce qu'il voit et des actions qu'il doit

entreprendre pour obtenir les résultats escomptés. Du point de vue du concepteur, une interface homme-machine est le logiciel et le matériel constituant le système informatique. La conception et la construction d'une interface sont des tâches difficiles et coûteuses. La difficulté provient de la nature même de l'interface qui requiert des compétences et des techniques multiples où interviennent l'informatique, l'ergonomie et la psychologie cognitive.

II-2 LES STYLES DE DIALOGUE

Il existe une analogie entre un dialogue homme-machine et un dialogue entre personnes. Après tout, les hommes ont développé au fil des années des moyens de communication assez sophistiqués et il serait important de tirer profit de toutes ces années d'expérience.

Plusieurs attributs du dialogue entre personnes doivent être conservés dans un dialogue homme-machine. En effet, les personnes qui communiquent ont acquis une certaine expérience et des connaissances qu'il faut représenter dans la machine. La tâche du concepteur est donc de garder les règles et le vocabulaire simple et d'essayer d'utiliser des concepts que l'utilisateur connaît déjà ou qu'il peut apprendre très vite.

Dans un dialogue entre personnes, on peut poser des questions, donner des réponses qui seront exprimées sous forme de mots, de gesticulations ou d'expressions faciales, en quelque sorte une image graphique. Une caractéristique d'un dialogue entre personnes qu'il est difficile de simuler dans un dialogue homme-machine est l'utilisation d'un contexte antérieur pour résoudre des ambiguïtés, ou une compréhension indirecte de référence. Ce problème commence à être résolu par des techniques d'intelligence artificielle.

En réalité, il y a deux langages dans une interface homme-machine: avec le premier, l'utilisateur communique avec l'ordinateur en s'exprimant par des actions appliquées aux différents outils; avec le second, l'ordinateur communique avec l'utilisateur en s'exprimant graphiquement à travers des lignes, des points, des chaînes de caractères, des zones de couleur combinées sous forme d'images et de messages.

Plusieurs styles de dialogue ont été développés pour les interfaces homme-machine. Nous présentons rapidement ici les plus importants.

1- **WYSIWYG**(What You See Is What You Get): c'est un système interactif graphique, où la représentation sur l'écran correspond à la réalité de l'application.

2- **Manipulation Directe**: dans la manipulation directe, les objets, les attributs et les relations sur lesquelles on peut effectuer des opérations sont visibles sur l'écran, ainsi que les opérations invoquées par l'application qui sont des actions sur les représentations visuelles. Il est certain que les interfaces basées sur cette technique comme le système Macintosh sont puissantes et faciles à apprendre.

3- **Langage de commande**: c'est la forme traditionnelle du dialogue avec l'ordinateur, elle est commode à utiliser, facilement extensible. Cependant, on fait plus d'erreurs avec les langages de commande qu'avec les menus.

4- **Langue naturelle**: si l'ordinateur arrive à comprendre des commandes tapées sur le clavier en langue naturelle ou la parole, toute personne est alors capable d'utiliser un ordinateur. Depuis de nombreuses années, le traitement du langage naturel fait l'objet d'un effort de recherche important. Pourtant, l'utilisation de la langue naturelle reste très limitée; cela provient sans doute du fait que durant trop longtemps, l'effort de la recherche a porté sur la reconnaissance et la compréhension de phrases. Or, pour permettre une utilisation effective de la langue naturelle dans une communication homme-machine, comprendre ou reconnaître une phrase ne suffit pas: il faut exploiter le résultat de cette compréhension dans le cadre d'un dialogue. Aujourd'hui, des systèmes robustes existent si l'on accepte de se limiter à des langages très contraints.

5- **Question/réponse**: le logiciel pose une série de questions auxquelles l'utilisateur répond. Il est complètement à l'initiative de l'ordinateur. Ce type de dialogue est probablement celui qui conduit le moins à des erreurs de la part des utilisateurs. Cependant, il devient vite lourd et ennuyeux à mesure que l'utilisateur acquiert de l'expérience.

Il est important de remarquer que ces styles de dialogue ne sont pas mutuellement exclusifs. En effet, plusieurs interfaces bien connues combinent différents styles de dialogue.

II-3 ERGONOMIE D'UNE INTERFACE HOMME-MACHINE

Concevoir une interface homme-machine n'est pas le simple "habillage" d'une application. Elle implique aussi bien des problèmes d'ergonomie, de modèle de représentation, que de méthodologie de conception et d'implémentation. L'aspect extérieur d'un logiciel est ce que l'utilisateur perçoit directement du système et qui conditionne son opinion aussi bien par le confort d'utilisation que par la facilité d'apprentissage et, par la suite, détermine l'acceptation du logiciel par l'utilisateur.

L'ergonomie d'une interface homme-machine étudie le comportement d'un utilisateur potentiel face à la création d'une interface et justifie les décisions du concepteur par rapport à l'activité des utilisateurs. En effet, il faut prendre en compte l'utilisateur dans les différents stades de la conception, à savoir l'étape d'analyse (spécification des contraintes, identification des performances d'usage et des caractéristiques de la population, analyse des tâches), et l'étape de conception d'interface.

Un certain nombre de travaux ont été effectués autour de l'ergonomie des interfaces. On peut citer notamment les travaux de *D. Scapin* à l'INRIA [Scap 86] dans "*le guide ergonomique de conception des interfaces homme-machine*", où l'auteur, d'une part décrit les défauts les plus courants des interfaces et d'autre part, donne des

recommandations. Parmi les défauts, il cite le fait d'aborder la réalisation selon une logique qui tient d'avantage compte des problèmes d'implémentation, comme les structures de fichiers, que de la façon dont l'utilisateur se représente l'information manipulée. Une autre lacune réside dans la méconnaissance de l'expérience préalable des utilisateurs.

J.D. Gould [Goul 88] a décrit dans son manuel *"How to design usable systems"* quatre principes à suivre pour concevoir des systèmes utilisateur et propose des méthodes pour appliquer ces principes. Cependant, les méthodes proposées sont empiriques et difficiles à appliquer pour la conception de gros systèmes, où la population des utilisateurs est très variée. Cela n'empêche pas que les méthodes proposées sont nécessaires mais pas suffisantes.

Enfin, *J. Coutaz* [Cout 86] a cité sept règles sous forme de conseils pratiques aux concepteurs d'interface homme-machine et montre la difficulté de créer une bonne interface, laquelle prend en compte les considérations ergonomiques.

II-3.1 Modèle utilisateur

La terminologie en ***user modeling*** est très confuse et ambiguë. Dans la littérature, plusieurs auteurs ont donné une définition plus ou moins précise du modèle utilisateur, du modèle mental et du modèle conceptuel. Cependant, il reste à définir le contexte exact de chaque catégorie pour éliminer toute confusion.

Les définitions suivantes m'ont paru les plus complètes et elles donnent une idée générale sur le modèle utilisateur.

> *"These models provide predictive and explanatory power for understanding the interaction. Mental modes evolve naturally through interaction with the world and with particular system under consideration.... these models are highly affected by the nature of the interaction, coupled with the person's prior knowledge and understanding."*
>
> ***Norman***

La définition de *Norman* est une des plus récentes, dans laquelle l'auteur voit le modèle conceptuel comme une forme mentale qui engloberait le modèle du concepteur et le modèle utilisateur.

> *"The representation of the user's knowledge of the system refering to the knowledge structures internal to the user rather to somme external reality. A user is a more or less organised body of knowledge, existing in the user's memory, which helps guide his or her own actions and aids interpretation of system actions."*
>
> ***Hammond***

Hammond définit le modèle utilisateur comme la représentation des connaissances, plus ou moins organisées, que l'utilisateur a du système d'où la difficulté du modèle utilisateur.

En résumé le modèle utilisateur est une représentation virtuelle de l'ordinateur qui doit coïncider avec la représentation mentale que l'utilisateur a du système. Or la modélisation de l'utilisateur impose à mon avis deux questions:

1- Comment peut-on comprendre et modéliser un utilisateur, plus encore, un ensemble d'utilisateurs allant du novice à l'expérimenté?

2- Comment appliquer cette compréhension et cette modélisation dans le développement du logiciel?

Ces deux questions touchent largement les issues pratiques et théoriques de la partie cognitive de l'informatique. Pour répondre à la première question, plusieurs approches ont été proposées dans la littérature, mais ces approches restent empiriques et ambiguës. On doit donc définir les différents groupes d'utilisateurs potentiels (du novice à l'expérimenté), et définir leurs besoins; ensuite, il faut évaluer leurs connaissances, leurs niveaux (c'est-à-dire, leurs capacités à assimiler et à comprendre des systèmes nouveaux) et ce qu'ils attendent du nouveau système. Cependant, la connaissance et les actions de l'utilisateur sont effectivement des connaissances intensives, complexes, difficiles à comprendre et parfois, pas ou difficilement "informatisables", ce qui rend le modèle utilisateur compliqué et difficile à mettre au point.

Actuellement, il n'existe pas, à ma connaissance, d'outils logiciel permettant l'évaluation automatique des connaissances ou des actions de l'utilisateur. En effet, la seule méthode d'évaluation des connaissances des utilisateurs, valable aujourd'hui, se fait à travers les interviews et l'observation des utilisateurs accomplissant leur tâche sans le système ou encore sur des système de simulation. La modélisation de l'utilisateur servira de guide pour le concepteur durant toute la conception de l'interface, à savoir la spécification, la conception, l'implémentation et l'évaluation.

Une fois la modélisation de l'utilisateur terminée (ce modèle n'est pas final et peut être modifier au cours de la conception), on doit intégrer ce modèle dans le système. Une approche classique consiste à tenir compte du modèle utilisateur dans chaque étape de la réalisation de l'interface. Pour les tâches utilisateurs, on doit établir le séquencement opérationnel entre événements, décisions et actions en utilisant les informations recensées dans le modèle utilisateur. De telles analyses permettent la construction d'organigrammes et de scénarios, qui seront les outils pour les décisions de la conception, les simulations et les évaluations. Ces scénarios doivent coïncider avec l'attente des utilisateurs. Ces analyses devraient faire partie d'un processus itératif au cours des différentes étapes de développement et d'implémentation du logiciel. Ce n'est qu'en étant une partie intégrante du processus de conception que ces analyses permettront d'éviter des erreurs de conception liées à une méconnaissance de la tâche et des utilisateurs, ou à un manque de méthodologie adaptée.

Une autre approche, plus récente, consiste à construire une base de connaissances sur les utilisateurs. Les informations requises sur les utilisateurs peuvent évoluer, un utilisateur novice peut évoluer en un utilisateur expérimenté. T. Fini et D Drager [Fini 86], proposent un modèle utilisateur généralisé appelé GUMS (General User Modelling Shell), qui peut être utilisé pour plusieurs applications, étant donné que dans plusieurs systèmes interactifs, le modèle utilisateur est valable pour une application particulière, ce qui est plus coûteux en temps et en efforts.

GUMS est conçu pour un ensemble d'applications; pour chacune, il maintient une base de connaissances des modèles utilisateurs. D'une part, les applications permettent d'acquérir des informations sur les utilisateurs, qui seront soumises à GUMS pour faire sa mise à jour. D'autre part, les applications reçoivent de GUMS des informations qui leur permettront de s'adapter aux profils de l'utilisateur (voir figure 15).

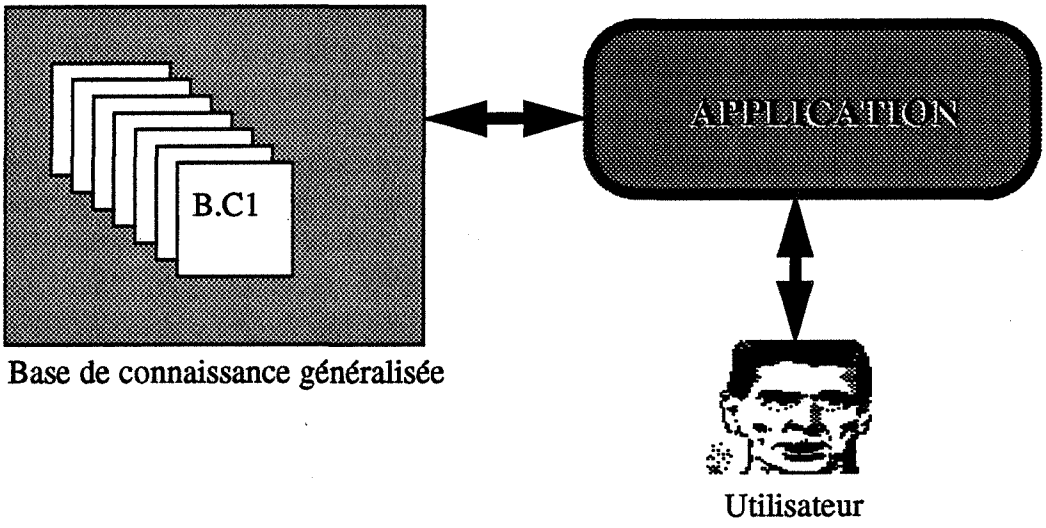


Figure 15: GUMS

Le modèle utilisateur est défini par des stéréotypes. Un stéréotype est un ensemble de "faits" et de "règles" qui définissent les caractéristiques d'une classe d'utilisateurs. Les stéréotypes ont une organisation hiérarchique où chaque stéréotype peut hériter d'un autre stéréotype plus général dans la hiérarchie. Une telle organisation peut être modélisée par objets où les utilisateurs individuels ne sont autres qu'une instance de classe d'utilisateurs.

Pour construire la base de connaissances, le problème d'acquisition d'informations sur les utilisateurs se pose. Deux approches sont possibles: l'acquisition explicite d'une part, mais cette approche est difficile à réaliser vu le nombre important d'informations concernant les utilisateurs et la difficulté de les classifier. Un autre inconvénient de cette approche est la non flexibilité de la base où les informations concernant les utilisateurs n'évoluent pas automatiquement. La deuxième approche est l'acquisition implicite de l'information, où le modèle est construit au fur et à mesure que l'utilisateur interagit avec

le système. Cependant, cette approche est lente pour construire un modèle utilisateur robuste et les conclusions de ce modèles sont incertaines. Il serait peut-être plus intéressant d'utiliser un mélange des deux approches, où un noyau de connaissances assez général est construit de façon explicite et où la base a la possibilité d'évoluer au fur et à mesure de l'interaction.

Il existe plusieurs techniques permettant la collection des données sur l'utilisateur; seulement peu sont "informatisables". En effet, le concepteur d'IHM à besoin d'outils permettant la collection des données et leur gestion. Les outils logiciels performants permettant le développement d'un modèle utilisateur sont rares sinon inexistants vu la complexité des facteurs humains. Néanmoins, je pense que les axes de recherche doivent porter sur les modèles généralisés tel que GUMS ou sur les outils permettant la vérification de la complétude et de l'exactitude du modèle utilisateur.

II-4 MODELISATION DE L'INTERFACE

La modélisation d'une interface homme-machine est une tâche difficile et demande beaucoup d'efforts. Quelques synthèses récentes sur le domaine ont été proposées, par exemple [Cout 88], [Myer 89] ou [Nana 90]. Parmi les modèles proposés, on peut distinguer:

- Les modèles d'analyse de tâches.
- Les modèles de dialogue.
- Les modèles d'architecture.

II-4.1 Les modèles d'analyse de tâches

Ces modèles sont destinés au concepteur d'applications auquel ils fournissent un schéma général du processus de conception. Le concept de tâche peut être défini comme la réalisation d'un but donné dans des conditions déterminées. Il est actuellement utilisé selon au moins deux points de vue dont la distinction est parfois floue: la modélisation du raisonnement et la modélisation d'une activité.

Dans le premier cas, il sert de moyen de structuration pour construire des systèmes de base de connaissances, appropriés à un type spécifique de résolution de problème. Chaque tâche peut utiliser des formes de connaissances et des stratégies qui lui sont propres.

Dans le second cas, il est utilisé pour modéliser l'activité de l'utilisateur dans un domaine d'application donné pour construire des systèmes. Les modèles de description des tâches sont alors utilisés pour représenter et planifier des activités, par exemple de bureau. Les modèles proposés, malgré certaines divergences, partagent un principe général commun: une tâche est une entité décrivant de manière déclarative les aspects fonctionnels et opérationnels de la tâche. Les aspects fonctionnels correspondent aux conditions d'exécution de la tâche et à ses effets. Les aspects opérationnels décrivent l'action ou les tâches à exécuter en réponse à un contexte fonctionnel donné [Nana 90].

Dans le domaine des interfaces, les modèles d'analyse de tâches sont utilisés pour analyser et décrire une tâche particulière de l'utilisateur, le plus souvent par décomposition hiérarchique en sous-tâches. Ils sont typiquement utilisés pour fournir un cadre de base au processus de conception d'une interface particulière. Ils identifient les types de connaissances que l'utilisateur a ou doit avoir pour réaliser sa tâche. Les modèles d'analyse de tâches au niveau le plus détaillé sont confrontés avec la prise en compte des détails de réalisation de la tâche qui sont dépendants du dialogue et des moyens matériels de sa réalisation. Dans cette catégorie de modèles, nous pouvons citer, à titre d'exemples, les modèles CLG [Mora 81] et MAD [Scap 89].

Les modèles de tâches apportent une contribution de valeur pour la conception d'interfaces: l'interface homme-machine doit s'appuyer sur la connaissance que l'utilisateur a de sa tâche. Ce point très positif n'occulte cependant pas qu'aucune aide n'est apportée sur la manière d'aborder la structuration de la tâche. Ces modèles ne permettent pas d'extraire la connaissance de l'utilisateur et de la structurer au fur et à mesure de son extraction. De plus, la correspondance entre les concepts et leurs représentations n'est pas mise en avant dans le processus de conception associé alors qu'elle est un point essentiel pour la compréhension de l'interface et de l'application par l'utilisateur.

II-4.2 Les modèles de dialogue

Ces modèles permettent au concepteur de dialogue de décrire la structure des échanges entre l'homme et l'application. Ils identifient les objets du dialogue. Green [Gree 86] définit un modèle de dialogue comme un modèle abstrait utilisé pour décrire la structure du dialogue entre l'utilisateur et un système interactif. On peut considérer qu'il existe deux types de dialogue: le dialogue séquentiel (modèle linguistique) et le dialogue asynchrone (modèle à événements).

II-4.2.1 Le modèle linguistique

Avec ce type de modèle, l'utilisateur décrit ce qui doit être fait par l'application ou par le système à l'aide d'un langage. Ce type de dialogue est de nature séquentielle dans la mesure où l'enchaînement de parties de dialogue est prévu à l'avance. Il n'est pas nécessaire qu'un tel dialogue repose sur une forme textuelle. En effet, ce dialogue peut être réalisé aussi bien sous forme de questions réponses, de commandes, que de navigation dans des réseaux de menus.

La difficulté d'utilisation de l'approche linguistique est la différence de nature qui existe généralement entre le langage de l'utilisateur (utilisé en entrée) et celui utilisé en sortie par l'application. La vue linguistique de l'interaction fait apparaître quatre niveaux:

- Le niveau lexical: il définit la forme externe des symboles utilisés, qui est dépendante du matériel et du style d'interaction.

- Le niveau syntaxique: il définit des séquences de symboles destinées à appeler des actions sémantiques aussi bien qu'à spécifier la forme et le contenu des sorties.

- Le niveau sémantique: il définit, du point de vue de l'application, des items d'entrée et de sortie qui ont un sens pour l'application.

- Le niveau conceptuel: il correspond à la collection des buts et fonctions de l'application que l'utilisateur final doit comprendre et attend du système.

Une des caractéristiques du modèle linguistique est qu'il considère un dialogue de nature séquentielle. L'ordre des lexèmes est imposé et correspond à la structure syntaxique du langage d'interaction. Ceci rend ce modèle inadapté à la modélisation de la manipulation directe.

II-4.2.2 Le modèle à événements

Ce modèle a émergé de la réalisation de systèmes graphiques qui considèrent les périphériques d'entrée comme des sources d'événements. Il est donc fondé sur la notion d'événement. Ces événements peuvent être produits soit par des périphériques, soit par l'interface elle-même, soit par l'application. A un événement est associé un nom ou un numéro permettant d'identifier la nature de l'interaction en plus des valeurs qui la caractérisent. Par exemple, dans le cas de la souris, un événement de déplacement est généré à chaque mouvement de celle-ci. Les valeurs associées à cet événement sont les coordonnées X et Y du curseur. Un autre événement est généré si le bouton de la souris est pressé.

Quand un événement est généré, il est envoyé à un ou plusieurs gestionnaires d'événements (event handler) qui sont susceptibles de le reconnaître, d'être intéressés par cet événement et de le traiter. Le traitement d'un événement déclenche une action qui peut provoquer l'exécution de fonctionnalités à l'intérieur de l'interface ou dans l'application, générer de nouveaux événements, créer ou détruire des gestionnaires d'événements.

Le modèle à événements est une extension de cette idée où un nombre arbitraire d'événements sont prédéfinis et le programmeur a la possibilité de définir de nouveaux types d'événements plus appropriés à une application particulière.

L'avantage du modèle à événements réside dans sa capacité à décrire un dialogue multifil. Dans un dialogue multifil, l'utilisateur peut maintenir plusieurs dialogues différents en parallèle. L'utilisateur a la possibilité de passer d'un dialogue à un autre à tout moment de l'interaction. Ce type d'interaction est rendu possible avec les systèmes de fenêtrage.

Les modèles de dialogue évoqués se dégagent mal du formalisme qui sert à les décrire ou de la réalisation qui en est faite dans les systèmes qui les utilisent. Le modèle linguistique de dialogue introduit une asymétrie entre les deux partenaires du dialogue. Le modèle à événements quant à lui, distingue mal le comportement interne du comportement externe du système.

II-4.3 Les modèles d'architecture

Ces modèles fournissent aux réalisateurs une structure générique à partir de laquelle il est possible de construire un système interactif particulier. Ils sont chargés de gérer effectivement la communication entre l'utilisateur et l'application et d'autre part de définir les composants de l'interface qui assurent cette traduction. Tous les modèles d'architecture mettent l'accent sur la séparation entre les aspects fonctionnels de l'application et les techniques de présentation et d'interaction.

On distingue différentes classes de modèles d'architecture:

- ***le modèle Entrées-Sorties***, qui correspond à une approche purement informatique de décomposition de l'interface en différents niveaux d'abstraction, chaque niveau correspondant à un aspect fonctionnel particulier. Les différents types de services généralement identifiés sont:

  - la gestion des unités physiques,
  - la gestion et le partage du poste de travail,
  - la gestion des événements abstraits (en entrée ou en sortie),
  - l'affichage et la désignation des représentations externes des entités de l'application,
  - la gestion du dialogue.

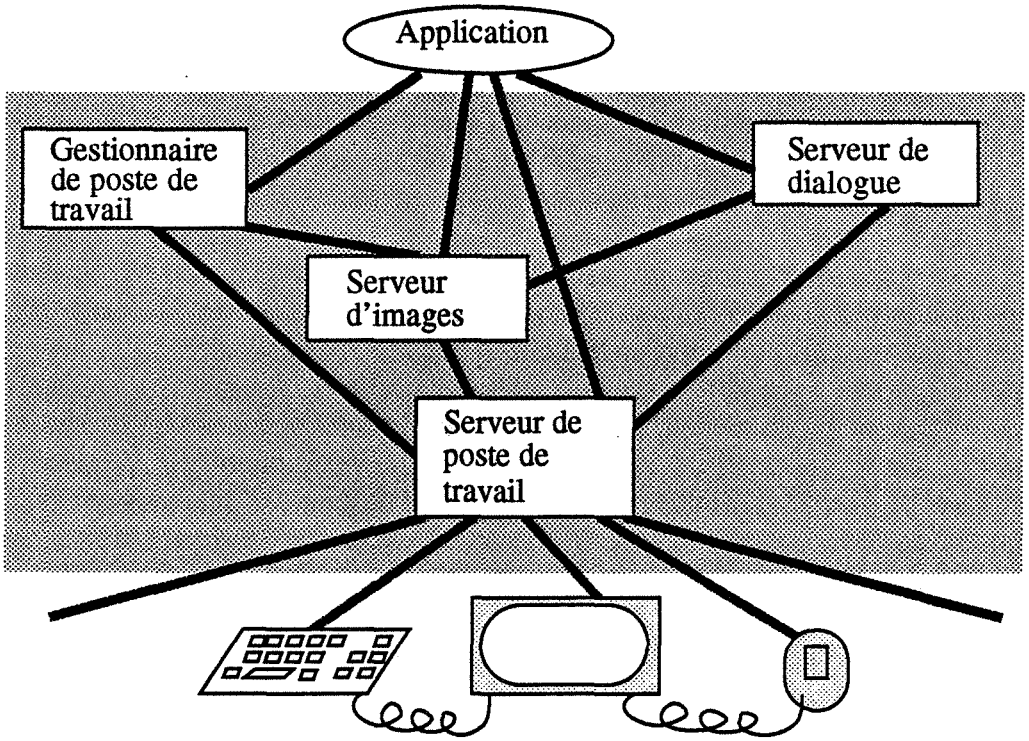


Figure 16: Vue modulaire du modèle Entrées-Sorties

Pour des raisons d'efficacité, ce modèle laisse la possibilité à l'application d'invoquer directement un niveau de service sans passer par les services intermédiaires.

- ***le modèle de Seeheim,*** qui correspond au modèle de dialogue linguistique. Il considère qu'un système interactif est constitué de deux partenaires (l'utilisateur et l'application) qui dialoguent par l'intermédiaire de l'interface. L'application est chargée des traitements sémantiques. L'interface se comporte comme un filtre avec trois composants:

  - le composant *Présentation* gère les aspects externes de l'interaction (niveau lexical),

  - le composant *Contrôleur* de dialogue a la charge de reconnaître la structure de l'interaction (niveau syntaxique); il contrôle en particulier les séquencements des éléments du dialogue.

  - le composant *Liaison* Interface-Application définit la vue que l'interface a de l'application et réciproquement la vue que l'application a de l'interface.

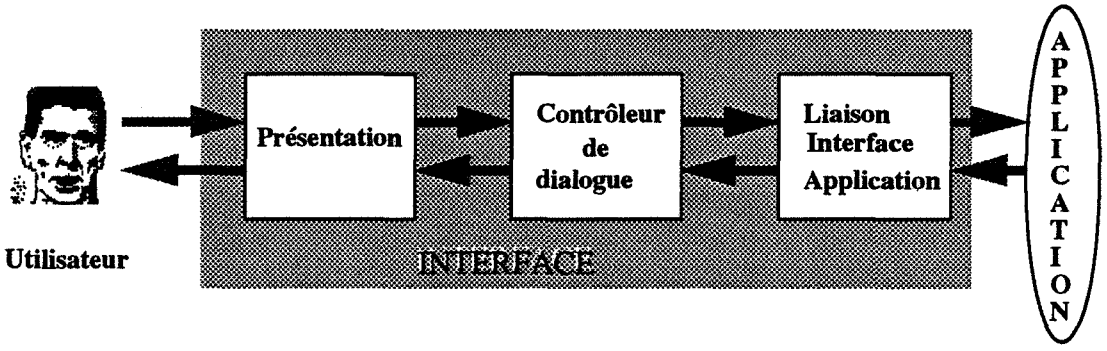


Figure 17: Le modèle de Seeheim

La communication entre ces trois composants se fait par échange de "tokens". L'interface peut donc considérer utilisateur et application comme deux systèmes réactifs dont elle contrôle les échanges.

- ***le modèle Multiagent,*** qui essaie de modéliser le fonctionnement d'un système comme un ensemble d'agents interagissant. Un système construit sur ce modèle est constitué d'un certain nombre d'agents spécialisés qui travaillent en parallèle et réagissent à des événements particuliers considérés comme des stimuli. Ils produisent en retour des événements qui sont des stimuli pour d'autres agents. Chaque agent est spécialisé dans la gestion d'un fil de dialogue. On peut citer à titre d'exemple le modèle PAC [Cout 88] qui s'appuie sur la notion d'agent.

Le modèle PAC structure récursivement un système interactif comme une

hiérarchie d'agents. Un agent encapsule une compétence correspondant à un certain niveau d'abstraction. Chaque agent présente trois facettes.

- La *Présentation* définit le comportement perceptible de l'agent: son image et les interactions qu'il est susceptible de recevoir.

- L'*Abstaction* définit les concepts associés à l'agent et les fonctionnalités assurées par l'agent relativement à ces concepts. Elle implémente donc une expertise et présente une interface abstraite vis-à-vis des autres agents.

- Le *Contrôle* maintient la cohérence entre les facettes *Présentation* et *Abstraction* et fait le lien avec les autres agents.

Un système interactif est un objet PAC composé. La figure18 tirée de [Cout 88] donne un exemple simplifié de structuration d'un système interactif. L'abstraction correspond à l'application. La Présentation définit l'image interactive du système. Le contrôle sert d'intermédiaire entre l'application et la présentation. Il met en correspondance les concepts de l'application avec les objets interactifs de la présentation modélisés par les objets PAC.

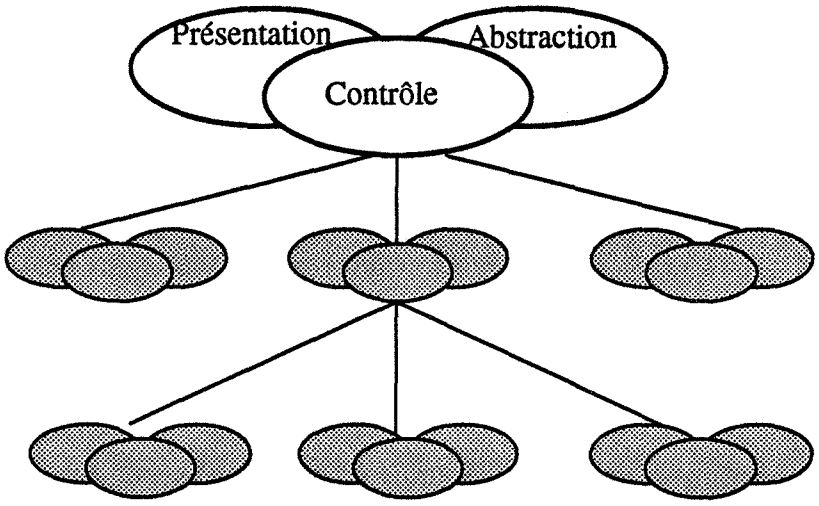


Figure 18: Structuration d'un système interactif en objets PAC

Comme le modèle de Seeheim, le modèle PAC met l'accent sur la séparation entre fontionnalités et techniques de présentation mais il introduit la notion de contrôle et la notion de répartition. La construction récursive d'un système interactif au moyen d'objets PAC fournit un moyen de structuration du système et permet de répartir le contrôle sémantique et syntaxique à différents niveaux d'abstraction.

La difficulté essentielle des modèles d'architecture concerne les échanges sémantiques entre la partie dialogue et la partie fonctionnelle. Le problème se complique

dans le cas d'applications à manipulation directe, dans la mesure où la représentation est le support de la sémantique perçue par l'utilisateur.

II-5 LES FORMALISMES DE REPRESENTATION

Un formalisme sert de support à la communication d'un modèle. Son choix est donc implicitement lié au modèle et dépend de son pouvoir expressif et de sa capacité à exprimer les divers aspects du modèle. La difficulté de choix provient de l'inaptitude des différents formalismes à représenter tous les aspects de l'interface. En effet, le problème de la complétude de la représentation de l'interface n'est pas encore résolu. C'est pourquoi le recours à la programmation traditionnelle vient encore compléter l'utilisation des méthodes formelles.

Les formalismes de représentation utilisés pour spécifier les divers aspects de l'interface peuvent être classés en plusieurs catégories, les plus importantes étant basées sur le diagramme de transition, les règles de production ou la grammaire hors-contexte.

II-5.1 Diagramme de transition

Un diagramme de transition est un graphe orienté formé d'un ensemble de nœuds et d'arcs. Chaque nœud correspond à un état du dialogue et un arc représente la transition entre deux états, commenté par les actions pouvant être réalisées à cette étape du dialogue (voir figure 19)

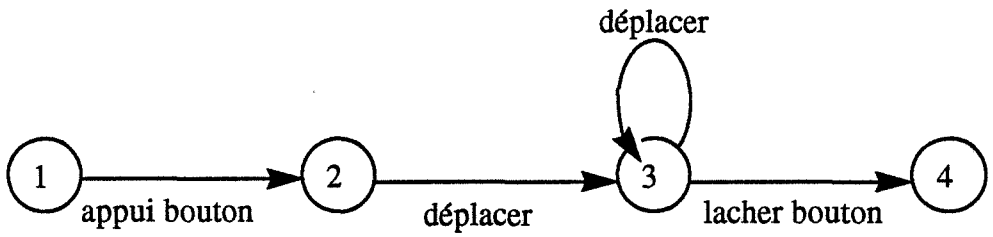
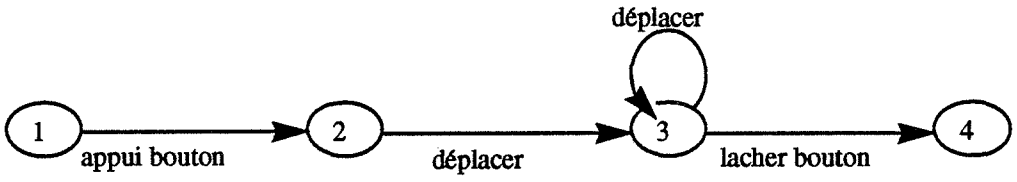


Figure19: Simple diagramme de transition

Un diagramme de transition présente l'avantage d'être facile à générer et à représenter graphiquement; de plus, l'information contenue dans ce diagramme est facile à saisir et à comprendre. On traduira automatiquement le graphe décrivant le schéma du dialogue en sous-programmes permettant son implémentation.

Les inconvénients majeurs de cette méthode sont: premièrement, le diagramme est trop long et il est nécessaire de le diviser en sous-diagrammes décrivant chacun une partie de l'interface. Deuxièmement, seul un sens du dialogue est traité: en effet, le diagramme décrit uniquement les actions de l'utilisateur mais non les réponses de l'application sur les actions. Une manière de décrire les réponses de l'application est d'attacher des actions aux

états du diagramme. Quand un état est atteint, son action est exécutée. Donc, les actions utilisateur sont attachées aux arcs et les actions de l'application sont attachées aux états. La figure 20 reprend l'exemple précédent en rajoutant les réponses de l'application.



action2= enregistrer le premier point
action3=tracer la ligne jusqu'à la position courante
action4= enregistrer le deuxième point

Figure 20: Diagramme de transition avec actions de l'application

Enfin, le diagramme de transition ne supporte pas les actions inattendues de l'utilisateur. En effet, quand le diagramme est dans un état particulier et que l'utilisateur effectue une action ne correspondant à aucun arc possible, le dialogue est bloqué. Pour cela, il existe plusieurs solutions: la plus simple est d'ignorer l'action de l'utilisateur. Cette solution n'est pas satisfaisante car l'utilisateur n'est pas informé que son action est ignorée et il ne sait pas comment passer à un autre état. Une autre solution consiste à utiliser un "jocker" qui sera emprunté s'il n'existe aucun arc satisfaisant l'action. Cet arc conduit à un état de recouvrement d'erreurs.

II-5.2 Les systèmes à base de règles de production

Le terme production vient du mécanisme qui consiste à produire des faits à partir d'autres faits. Chaque règle de production se présente sous la forme:

*Si <prémisse> **alors** <conséquence>*

Lorsque la prémisse (ensemble de conditions) est vérifiée, alors la conséquence est déclenchée. Elle correspond à un ensemble de conclusions qui sont affirmées comme vraies et/ou un ensemble d'actions qui sont déclenchées. Dans ce mode de représentation, les régles sont indépendantes, ce qui introduit une situation de non déterminisme résolu par un contrôleur de stratégie qui gère le déclenchement des règles. Ce type de représentation est à la fois déclaratif et procédural [Nana 90].

Le formalisme des règles de production est utilisé pour décrire le modèle de dialogue à événements pour la génération d'interfaces par plusieurs systèmes. Une utilisation du formalisme des règles de production est rencontrée pour la spécification des "View Controlers" qui constituent le contrôleur de dialogue du système Serpent [Bass 88].

II-5.3 Les systèmes de production de grammaire

Le dialogue homme-machine peut se traduire sous forme de grammaires BNF. Nous ne considérerons ici que les grammaires dites hors-contexte. En pratique, une grammaire est utilisée afin de décrire le langage d'un utilisateur lui permettant de communiquer avec l'ordinateur. Dans l'autre sens (ordinateur-homme), la description peut être faite par une autre méthode. Un terminal dans la grammaire est l'unité syntaxique représentant l'action utilisateur. Les terminaux sont combinés avec les productions de la grammaire pour former les non terminaux. Une collection de productions de la grammaire définit le langage employé par l'utilisateur dans son interaction avec l'ordinateur. L'exemple de la figure 20 se traduit ici par:

Ligne::= bouton, point
point::= déplacer, point | bouton

Dans le cas du diagramme de transition, les actions de l'application sont attachées aux états du diagramme. Une approche similaire peut être utilisée avec les grammaires hors-contexte; les actions de l'application peuvent être attachées à chaque production de la grammaire. Ces actions sont exécutées lors de l'analyse des entrées utilisateur. Le problème avec cette approche est que les exécutions des productions dépendent de l'algorithme utilisé (algorithme ascendant ou descendant). L'exemple précédent peut être formulé comme suit.

Ligne::= bouton, d1, point
point::= déplacer, d2, point | bouton d3

d1::={ enregistrer le premier point}
d2::= {tracer la ligne jusqu'à la position courante}
d3::= {enregistrer le deuxième point}

Actuellement, il existe sur le marché des outils qui génèrent automatiquement des interfaces utilisateur à partir de la définition de la grammaire. Par exemple Syngraph (Syntax directed Graphics) [Olse 83] est un système qui génère une interface utilisateur en Pascal à partir d'une description en grammaire formelle (Une extension de BNF). Syngraph permet la création de menus, la saisie de texte et l'utilisation de différents outils d'interaction. La grammaire est utilisée pour produire un analyseur de dialogue en mode descendant (Top-down) pour décrire quelques informations nécessaires à la représentation. Ces informations décrivent le contenu des menus et quand tel ou tel outil doit être utilisé.

Cependant, Syngraph ne permet pas de faire le retour arrière sémantique puisqu'il n'existe aucun moyen pour l'application de modifier le comportement de l'analyseur. Une telle grammaire ne peut être utilisée que si le dialogue dépend étroitement du contexte.

II-6 OUTILS DE DEVELOPPEMENT D'INTERFACES

On ne peut connaître le bon fonctionnement d'une interface qu'après avoir fait des essais. Ceci nous conduit à la construction rapide d'un prototype et à l'itération de la modélisation, d'où la nécessité d'avoir des outils d'aide à la création d'interfaces utilisateur.

Les outils actuellement disponibles pour le développement d'interfaces peuvent être classés en trois catégories principales:

- Les boîtes à outils,

- Les squelettes d'application,

- Les systèmes de génération d'interfaces.

II-6.1 Les boîtes à outils

Une boîte à outils est une bibliothèque de procédures pour le développement d'interfaces utilisateur. L'éventail des fonctions offertes va de la gestions des événements physiques de bas niveau tels que le clic de la souris et l'affichage d'un point, à la gestion d'entités de dialogue de haut niveau tels que les menus et les ascenseurs. Ces fonctions peuvent être classées en deux catégories:

1- les fonctions de gestion du poste de travail: elles définissent un appareil logique dont la fonction est de cacher le fonctionnement de l'appareil physique.

2- les fonctions de gestion de dialogue: Ces fonctions s'appuient sur les fonctionnalités du niveau inférieur décrit précédemment. On y trouve des entités comme les icônes, les menus, les boutons, les formulaires.

La boîte à outils (Toolbox) de Macintosh constitue une bibliothèque qui peut être vue comme la juxtaposition de boîtes à outils élémentaires, chacune fournissant un sous-ensemble de procédures spécifiques d'un problème donné. Citons par exemple: le graphique, les polices de caractères, les événements, les fenêtres, les menus,... L'idée est de présenter au développeur des fonctionnalités pré-empaquetées qui masquent les couches basses de réalisation et assurent ainsi une certaine sécurité en fixant des conventions.

Les boîtes à outils ont plusieurs avantages dont les plus importants sont la flexibilité, l'extensibilité et une plus grande portabilité des applications interactives.

Les inconvénients des boîtes à outils sont nombreux. En voici un résumé:

- une mauvaise décomposition modulaire qui remet en cause l'aspect itératif de la mise au point du système interactif (conséquence de la souplesse);

- une difficulté d'apprentissage due à l'aspect "en vrac" des primitives de la boîte. D'après [Cout 88], trois mois sont nécessaires pour maîtriser une boîte à outils;

- spécificité de l'interface utilisateur vis-à-vis des applications de l'utilisateur;

- une existence de plusieurs boîtes à outils sur le marché, ce qui gêne la portabilité des logiciels interactifs;

-une duplication d'efforts: un style de programmation identique pour plusieurs applications.

II-6.2 Les squelettes d'application

Un squelette d'application est une structure de contrôle partagée par des applications, qui devient ainsi réutilisable. Il regroupe sous la forme d'une architecture réutilisable et extensible, un noyau d'exécution et un ensemble de services communs à un ensemble de systèmes. Le développement d'une application consiste à compléter les trous du squelette et à redéfinir les parties prédéfinies inadaptées au cas particulier de l'application. Un squelette met donc à disposition du programmeur une architecture prête à l'emploi.

APEX [Cout 88] est un exemple d'un squelette d'application organisé selon le modèle PAC, au-dessus de la Toolbox du Macintosh. La partie Présentation contient les classes d'objets interactifs réutilisables à partir desquelles sont instanciés les objets de l'interface. La partie Abstraction est structurée en deux niveaux: une zone de communication et la partie fonctionnelle de l'application. La partie contrôle constitue le noyau d'exécution.

Cette approche réduit sensiblement les coûts de développement et conduit à un style d'interaction cohérent. Cependant, les inconvénients de cette approche sont: premièrement, le niveau d'abstraction de la programmation de l'interface est trop bas et le style d'interaction est limité à ceux qui existent déjà, ce qui contraint le concepteur à rechercher dans la bibliothèque les fonctions nécessaires à l'extension ou à la modification du squelette de base. Deuxièmement, les bibliothèques sont difficiles à développer car on utilise généralement des langages de programmation textuels. Enfin, elles sont souvent étroitement liées à la configuration matérielle de la station sur laquelle elles sont implémentées.

II-6.3 Systèmes de génération d'interfaces (SGI)

La définition d'un système générique est ambiguë et les spécialistes ne sont généralement pas d'accord sur la définition d'un SGI. Pour certains, c'est un ensemble d'outils permettant la génération automatique d'applications interactives exécutables à partir de spécifications de haut niveau [Cout 86]. Pour d'autres, c'est un ensemble de services de présentation de données ou d'informations à l'utilisateur et l'interprétation des actions utilisateur à l'application. Elle fait le lien entre une variété d'utilisateurs et une

variété d'applications [Benn 86]. Cependant, une définition plus générale considère le SGI comme un ensemble d'outils qui aide à créer et à gérer tous les aspects de l'interface utilisateur, pendant sa création et son utilisation.

Les systèmes de génération d'interfaces engendrent une application interactive à partir d'une spécification de haut niveau. Il comprend un noyau d'exécution et des outils de spécification. A chaque application correspond le résultat d'une spécification. La spécification définit la syntaxe concrète du dialogue et la correspondance entre celle-ci et la syntaxe abstraite. La syntaxe concrète définit le mode d'emploi de l'application tel que le perçoit l'utilisateur. La syntaxe abstraite définit le format des informations échangées à l'exécution entre le noyau d'exécution et l'application [Cout 86].

Un SGI devrait fournir les mécanismes permettant de suivre le cycle de vie d'un système interactif depuis sa conception, jusqu'à sa génération, sa mise au point et sa maintenance. On peut classer les SGI en trois catégories:

- Les systèmes de génération d'interfaces basés sur un langage

- Les systèmes à manipulation directe de génération d'interfaces

- Les systèmes de génération d'interface à partir de spécifications fonctionnelles

II-6.3.1 Les systèmes de génération d'interfaces basés sur un langage

Cette catégorie de SGI se caractérise par une spécification souvent écrite. L'architecture de l'interface est largement influencée par le type de formalisme utilisé pour la spécification, et donc le modèle de dialogue implicitement utilisé. Cette approche est à la limite de la programmation, utilisant soit un langage dérivé d'un langage de programmation (par exemple Pascal), soit un langage spécialisé.

II-6.3.2 Les systèmes de génération d'interfaces à partir de spécifications fonctionnelles

L'objectif de ces systèmes est de générer l'interface à partir d'une description des propriétés sémantiques de l'interface. Le concepteur n'aurait plus qu'à modifier l'interface pour l'améliorer en s'aidant éventuellement d'un évaluateur. Un des systèmes de cette catégorie est UIDE [Foly 87] utilisant le langage IDL pour spécifier le niveau sémantique. Le niveau syntaxique et le niveau lexical sont générés automatiquement.

Cette approche est séduisante pour le concepteur qui se trouve libéré des détails de l'interaction. Cependant, les stratégies utilisées pour la production automatique de la représentation ne sont pas assez fines au regard de la manipulation directe.

II-5.3.3 Les systèmes à manipulation directe de génération d'interfaces

Cette catégorie de systèmes se caractérise par le fait que le concepteur manipule directement les objets graphiques qui seront en contact avec l'utilisateur final du système. Le concepteur a donc la possibilité de tester directement par lui-même les effets de ses choix. Deux classes de système se distinguent selon le type de fonctionnalités offertes. Dans la première, le concepteur spécifie l'interface en plaçant sur l'écran les objets avec lesquels l'utilisateur final interagira. Ces objets sont instanciés à partir d'une base d'objets prédéfinis, qui ont donc une représentation et un type de comportement prédéfinis. De ce fait, les interfaces produites sont stéréotypées. Il s'agit en fait d'assembler des objets sous forme de tableaux de bord, de spécifier l'enchaînement des tableaux de bord en précisant les effets des interactions de l'utilisateur, de préciser la liaison entre les objets de l'interface et les concepts de l'application. Ces systèmes sont limités à des objets prédéfinis. En effet, sans le recours à la programmation, il est difficile de spécifier des objets ayant un comportement et une présentation personnalisés.

Un système représentatif de la deuxième classe est Peridot [Myer 87]. L'objectif de Peridot est justement de permettre la définition de nouvelles techniques d'interaction par la manipulation directe pour la création d'interfaces. L'originalité de ce système est de permettre au concepteur de spécifier son interface par démonstration.Pour construire de nouveaux objets d'interaction (menus, barres de défilement,...), le concepteur dessine sur l'écran la représentation de ces objets (celle que verra l'utilisateur) en manipulant des figures géométriques primitives (rectangle, cercle, texte,...) et montre alors les actions que pourra faire l'utilisateur en mimant lui-même ces actions avec la souris. En utilisant une base de règles exprimant des relations géométriques possibles entre les objets, le système infère à partir des actions du concepteur les propriétés géométriques générales des objets (centrage, alignement,...) et applique ces règles de construction découvertes pour présenter au concepteur l'objet qui correspond à ses intentions.

La faiblesse de Peridot est de ne pas disposer d'un mécanisme d'apprentissage pour enrichir la base de connaissances. Néanmoins, l'idée est séduisante, non seulement pour aider à la conception d'interfaces à manipulation directe et tester immédiatement une présentation ou un mode d'interaction, mais aussi comme principe général d'assistance à un utilisateur.

II-7 LES INTERFACES MULTI-MODES

Il est incontestable que la technologie des interfaces prend de plus en plus d'importance dans la conception des systèmes informatiques. Des interfaces utilisant un seul mode de communication ne sont pas totalement satisfaisantes. Pour améliorer la qualité du dialogue entre l'utilisateur et l'application, il faut conserver les habitudes de l'utilisateur en mixant différents modes de dialogue. En effet, dans un dialogue entre humains, ceux-ci utilisent la langue naturelle, le graphique et parfois des expressions faciales et des gestes. Il est donc important qu'une interface puisse comprendre et générer ces différents modes de communication; ainsi, l'interface doit choisir le mode le plus approprié pour formuler un acte de communication.

Il est important de faire la distinction entre un système multimédia et un système multi-mode (multi-modal). En effet, un système informatique multimédia est capable d'acquérir, de restituer, de mémoriser et d'organiser des informations de nature écrite, sonore ou visuelle, sans pour autant être capable de comprendre leur contenu sémantique. Il doit pour cela être équipé d'équipements matériel et logiciel lui permettant l'acquisition, la restitution, le stockage et la manipulation de chaque type d'information. Par contre, un système informatique multi-mode (multimodal) est capable de communiquer avec ses utilisateurs en respectant les modalités de communication humaine. Il doit pour cela être équipé du matériel et des logiciels adaptés à l'acquisition, à la restitution et à la compréhension d'énoncés multimodaux [Cout 91]. Cette compréhension implique la traduction des énoncés en une représentation abstraite qui constitue la connaissance sémantique des énoncés, et d'effectuer des raisonnements sur cette connaissance en tenant compte du contexte du dialogue. La restitution s'effectue selon le processus inverse.

Certains systèmes multi-modes, bien qu'ils disposent de plusieurs modes de communication, ne peuvent pas employer simultanément plusieurs modalités. A l'inverse, des système tels que IPCdraw [Cael 91] et MMI2 permettent à l'utilisateur de formuler un énoncé en utilisant indifféremment l'un ou l'autre mode de communication. En conséquence, comme l'a précisé Coutaz [Cout 91], il faut distinguer la cohabitation de modalités et l'usage simultané de modalités:

- dans un système de type cohabitation, plusieurs modalités de communication sont disponibles mais leur usage s'effectue en exclusion ; la forme d'un énoncé utilise une seule modalité;

- dans un système de type usage simultané, la forme de l'énnoncé peut faire intervenir plusieurs modalités.

De plus, dans les systèmes multi-mode, il ne suffit pas de juxtaposer plusieurs modes mais il faut les faire coopérer au cours d'un même échange avec la machine. L'utilisateur et la machine doivent pouvoir exprimer un énoncé à travers plusieurs modes de communication: par exemple, pour demander à l'application de déplacer un objet sur l'écran, l'utilisateur peut taper la commande DEPLACE sur le clavier et désigner l'objet et sa nouvelle position avec la souris. La mise en œuvre de telles interactions nécessite la conception de nouvelles techniques telle que la gestion des événements multimodaux. Dans un sytème multi-mode il n'est pas suffisant de gérer les événements comme dans un système mono-mode où les événements sont stockés dans une pile FIFO en attendant leur traitement; il faut combiner plusieurs événements d'origine différente pour restituer un message entier.

Cette approche implique des problèmes de synchronisation. L'exemple présenté par Caelen [Cael 91] décrit parfaitement la problématique: si l'utilisateur déplace le triangle du coin gauche de l'écran pendant qu'il donne l'ordre de "changer la couleur du triangle du coin gauche" en utilisant la parole comme mode de communication, il est évident que lorsque la commande sera reconnue (les temps de réponse des différents média sont

différents) le triangle désigné sera ailleurs. La solution à ce problème, proposée dans le système IPCdraw [Cael 91], est basée sur l'utilisation d'un tableau noir (black-boad) qui mémorise le type, la provenance, la chronologie et la durée de chaque événement. Une boucle d'attente permet de gérer ce tableau noir.

Une interface multi-mode doit être intelligente et fournir une assistance coopérative entre l'application proprement dite et l'utilisateur. Pour cela, elle doit avoir des connaissances diverses concernant l'utilisateur (niveau d'expertise, caractéristiques,..), l'application (domaine de la tâche) , et la manière de conduire un dialogue intelligent (principe de négociation, de suggestion,...).

Hayes [Haye 86] a indiqué quels étaient les principes nécessaires à la construction d'une interface multi-mode avec une base de connaissances: l'utilisation du graphique permet de donner une vue sur le domaine d'application de la base de connaissances et un dialogue en langue naturelle permet à l'utilisateur d'exprimer ses idées de façon plus simple et parfois plus aisée qu'avec le mode graphique. Ce dernier permettra à l'utilisateur de référencer des entités en les désignant avec la souris, au lieu de les décrire de manière textuelle.

Plusieurs systèmes multi-modes avec une base de connaissance ont vu le jour ces dernières années. On peut citer, à titre d'exemple, les projets ACCORD [Kris 89], LOCKHEAD [Tyle 88], GRADIENT [Holl 87], ISIS [Maso 88], EUROHELP [Hans 88] et XTRA [Allg 89]. Dans ce qui suit, nous présenterons rapidement le projet ACCORD qui a des similitudes avec le projet MMI2.

ACCORD

Accord [Kris 89] est un projet Esprit qui intègre la langue naturelle (l'anglais, le français et l'allemand) et le graphique pour l'interrogation et la mise à jour d'une base de connaissances. Le domaine de l'application est le transport routier. Du point de vue graphique, l'application utilise des plans géographiques contenant les principales villes et le reseau routier. Des denrées alimentaires sont stockées dans les villes et différents moyens de transport permettent d'acheminer les denrées d'une ville à une autre. L'objectif du système est de trouver un véhicule qui convienne à ce type de transport et qui soit disponible, et de déterminer le chemin optimal que doit suivre le véhicule. D'autres informations sont associées à des entités graphiques pouvant être représentées sous forme d'histogrammes, de camemberts, de courbes,..

Le noyau du système est le gestionnaire de dialogue, qui contrôle les interactions utilisateur et la communication entre les différents modules du système (l'analyseur, le système graphique, la base de connaissances, le générateur de texte). Tous ces modules communiquent entre eux à travers un langage de représentation sémantique baptisé InL. InL est une extension de la logique du premier ordre indexée.

Le projet s'est heurté à un problème dû au fait que les objets graphiques sont traités comme des noms. Cette solution est appropriée à la langue naturelle mais semble inadaptée

au graphique, pour plusieurs raisons. Premièrement, dans le système graphique, les objets sont des entités et non des représentations graphiques des objets du domaine. Deuxièmement, InL tel qu'il a été défini ne permet pas d'exprimer les actions graphiques de la même façon que les actions en langue naturelle.

II-8 CONCLUSION

Le caractère empirique de la conception des interfaces comme toute activité de conception implique un processus de raffinement. Les travaux pour évaluer les interfaces s'appliquent à des niveaux assez divers. Ils concernent surtout l'évaluation a posteriori. Par ailleurs, les outils disponibles pour produire effectivement des interfaces nécessitent encore de la part du concepteur un lourd travail de développement et de mise au point. Pour pallier ces inconvénients et éviter des investissements importants de développement qui deviendraient inutiles en raison des choix de conception qui s'avéreraient erronés, le prototypage rapide des interfaces est devenu un besoin.

Des outils qui permettent un prototypage rapide sont très utiles sinon nécessaires dans la construction d'une interface homme-machine car ils offrent une version tangible et exécutable du système final. Un prototype permet de tester la flexibilité de la modélisation; il détermine aussi si le système satisfait les objectifs avant de faire l'implémentation complète.

Les éditeurs générateurs interactifs d'interface trouvent là leur principal intérêt, même si actuellement la puissance des interfaces générées est encore limitée. Ils incorporent directement des facilités de visualisation des interfaces produites. Un système comme Hypercard qui intègre au sein d'une même interface la définition et l'utilisation du système se révèle typiquement adapté pour une activité de prototypage rapide.

CHAPITRE III

LES CARTES PLANAIRES

Ce chapitre rappelle les définitions d'une carte planaire et présente un algorithme de construction incrémental d'une carte planaire. Il évoque les inconvénients de telles structures de données dans un environnement interactif qui sont dus essentiellement à l'imprécision numérique

III- LES CARTES PLANAIRES

III-1 INTRODUCTION

L'analyse de la tâche de conception de réseaux informatiques [Caho 91] a montré que les utilisateurs ont souvent besoin d'un support graphique pour accomplir cette tâche. De plus, au cours de la conception, ils peuvent avoir recours à des actions graphiques, par exemple dessiner une station ou un câble. Les objets graphiques manipulés par les utilisateurs peuvent être classés en deux catégories:

- les objets du bâtiment: ce sont les murs, les pièces et les étages du bâtiment dans lequel on veut installer le réseau; ils sont représentés sous la forme de droites en 2D ou en 3D.
- les objets du réseau: ce sont les composants matériels et logiciels d'un réseau informatique. Certains de ces objets sont représentés par des droites, par exemple les câbles, d'autres ont une représentation icônique (serveur, multiplexeur, ...).

En conséquence, l'outil graphique doit fournir les moyens nécessaires à l'utilisateur et à l'application pour manipuler les objets graphiques du bâtiment et du réseau. De plus, il doit être capable d'extraire la sémantique du tracé. Pour cela, il est nécessaire d'avoir un modèle adéquat pour représenter les objets à l'intérieur de l'outil graphique.

Dans la modélisation d'objets graphiques par les frontières en 2D et 3D, on fait de plus en plus appel au concept de carte. En effet, celui-ci procure un cadre mathématique rigoureux permettant de bien appréhender certaines notions topologiques et/ou géométrique: sommet, arête, brin et surtout bord (ou face). Il fournit en outre une masse d'informations théoriques permettant d'améliorer les algorithmes de manipulation de ces objets.

De nombreux travaux ont été réalisés sur les cartes planaires, notamment dans un contexte 2D [Mich 87],[Gang 89] et plus récemment dans un contexte 3D [Lien 88]. Tous ces travaux insistent d'une part sur la séparation entre les aspects topologiques (concernant les relations existant entre les différents éléments de l'objet modélisé: sommets, arêtes et bords), et les aspects géométriques, d'autre part sur le fait que la topologie doit être cohérente avec la géométrie.

En modélisation par les bords, un solide est défini par une subdivision d'une surface orientable fermée. Une telle subdivision, plongée dans l'espace tridimensionnel, partitionne cet espace en deux régions distinctes: l'une de ces régions définit le volume extérieur de l'objet modélisé, l'autre définit le volume intérieur à l'objet. De nombreux travaux ont porté sur la définition de modèles informatiques permettant la représentation de la subdivision de la surface et privilégiant l'information topologique. Ces travaux portent aussi sur la définition d'opérations élémentaires (opérations euclidiennes)

applicables à ces modèles [Lien 89].

Dans ce chapitre, nous étudions les cartes planaires (de dimension 2) permettant la subdivision de surfaces. La sections 2 présente un rappel théorique sur les cartes planaires en faisant une distinction entre l'aspect topologique et l'aspect géométrique des cartes planaires.

La section 3 expose la construction non-incrémentale d'une carte planaire, c'est-à-dire la construction de la carte planaire à partir d'un ensemble de points déjà saisi. Cette construction s'effectue en trois étapes; la première est une étape d'initialisation. La deuxième étape utilise l'algorithme de Bentley-Ottman [Bent 79]pour déterminer les points d'intersection entre les segments. La troisième étape construit l'arbre d'inclusion par un parcours de la structure de données; cet algorithme est dû à Gangnet et Michelucci [Mich 84].

La section 4 présente la construction des cartes planaires incrémentales, c'est-à-dire l'insertion et la suppression dynamique de segments, ainsi que le calcul dynamique des nouveaux points d'intersection et la mise à jour des structures de données.

Enfin nous conclurons dans la section 5.

III-2 RAPPELS SUR LES CARTES COMBINATOIRES

III-2.1 Les cartes topologiques

1 Carte et brin

On appelle carte, un triplet G=(B,α,σ) où:
- B est un ensemble fini dont les éléments sont appelés brins,
- α une involution sans point fixe de B,($\forall$ b $\in$ B (α^2(b)=Id(b) et α(b)≠b))
- σ est une permutation de B.

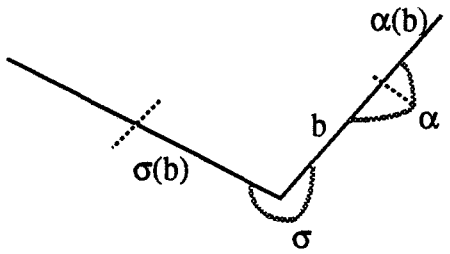


Figure 21: brins

NB: On note Aτ* l'orbite {τ^p(b), b$\in$ A, p≥0, A$\subseteq$B}, c'est à dire l'ensemble des brins de B accessibles par τ à partir de ceux de A.

2- Arête et sommet

Soit la carte G=(B,α, σ). a=bα*= {b,α(b)} où b ∈ B, est appelé arête de la carte G; b et α(b) sont dit opposés. Pour un brin b, on notera également -b le brin α(b).

s= bσ* = {b, σ(b), σ^2(b),.....,σ^{k-1}(b)} où k est le plus petit entier naturel non nul tel que σ^k(b)=b est appelé sommet de G. Les brins b et σ(b) sont dits adjacents.

3- Bord

On appelle bord direct (resp. bord rétrograde) l'ensemble {b_{i1},b_{i2}..... b_{in}} de G tel que:

∀ k,∃j, $b_{ik}=\sigma^{-1}$o α (b_{ij}); (resp. σ o α (b_{ij})).

Un bord parcouru dans le sens des aiguilles d'une montre est appelé bord externe; le bord est appelé bord interne sinon.

Propriétés:

♦ Si *f* est un bord interne, il existe $b \in f$ tel que $\alpha(b) \notin f$.

♦ Un bord interne *f* est orienté dans le même sens que α, c'est-à-dire dans le sens trigonométrique, ce qui équivaut à dire que lors du parcours du bord, son intérieur se trouve à gauche.

♦ Une composante connexe d'une carte planaire comporte un et un seul bord externe.

♦ Une arête {b,α(b)} appartient soit à deux bords internes distincts ou confondus, soit à un bord interne et à un bord externe, soit à un et un seul bord externe.

Exemple:

considérons la carte définie par la figure 22

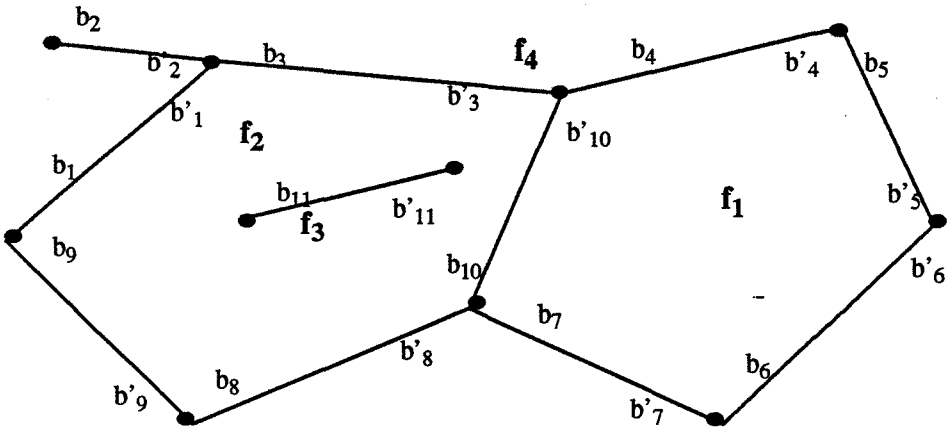


Figure 22: une carte planaire

arêtes:

$a_1=\{b_1,b'_1\}$; $a_2=\{b_2,b'_2\}$; $a_3=\{b_3,b'_3\}$; $a_4=\{b_4,b'_4\}$; $a_5=\{b_5, b'_5\}$; $a_6=\{b_6,b'_6\}$; $a_7=\{b_7,b'_7\}$; $a_8=\{b_8,b'_8\}$; $a_9=\{b_9,b'_9\}$; $a_{10}=\{b_{10},b'_{10}\}$; $a_{11}=\{b_{11},b'_{11}\}$.

sommets:

$s_1=\{b_9,b_1\}$; $s_2=\{b_2\}$; $s_3=\{b_8,b'_9\}$; $s_4=\{b_3,b'_2,b'_1\}$; $s_5=\{b_{11}\}$; $s_6=\{b'_{11}\}$; $s_7=\{b_7,b_{10},b'_8\}$ $s_8=\{b_4,b'_3,b'_{10}\}$; $s_9=\{b_6,b'_7\}$; $s_{10}=\{b_5,b'_4\}$; $s_{11}=\{b'_5,b'_6\}$.

bords:

$f_1=\{b_7,b_6,b'_5,b'_4,b'_{10}\}$ est interne;
$f_2=\{b_9,b_8,b_{10},b'_3,b'_1\}$ est interne;
$f_3=\{b_{11},b'_{11}\}$ est externe;
$f_4=\{b_1,b'_2,b_2,b_3,b_4,b_5,b'_6,b'_7,b'_8,b'_9\}$ est externe.

4- Sous-carte et carte partielle

Une carte $G'=(B',\alpha',\sigma')$ est appelée sous-carte de $G=(B,\alpha,\sigma)$ si:
(i)$B' \subseteq B$;
(ii)$B= B- (B-B'\sigma^*)\alpha^*$ et $B'\alpha= B'$;
(iii) $\alpha'=\alpha|B'$ et $\sigma'=\sigma|B'$ où σ' est tel que $\forall b\in B'$, $\sigma'(b)=\sigma^k(b)$ et k est le plus petit entier non nul tel que $\sigma^k(b)\in B'$.

autrement dit, G' est obtenu en supprimant certains sommets de la carte G, ainsi que les arêtes issues de ces sommets.

Une carte $G'=(B',\alpha',\sigma')$ est appelée carte partielle de $G=(B,\alpha,\sigma)$ si:
(i) $B'\subseteq B$;
(ii)$B'\sigma^*= B$ $(= \{\sigma^p(b); b\in B', p\geq 0\})$ et $B'\alpha= B'$;
(iii) $\alpha'=\alpha|B'$ et $\sigma'=\sigma|B'$;

autrement dit, G' est obtenu en supprimant certaines arêtes de la carte G, tout en conservant les mêmes sommets.

5- Classification des arêtes

☛ *arête pendante:*

Une arête $a= \{b, \alpha(b)\}$ est dite pendante si b et $\alpha(b)$ appartiennent au même bord.

☛ *arête de connexion:*

Une arête $a= \{b, \alpha(b)\}$ est dite de connexion si elle est pendante, si $\alpha(b)\neq b$ et si $\sigma(\alpha(b))\neq\alpha(b)$.

☛ *arête terminale:*

Une arête a= {b, α(b)} est dite terminale si elle est pendante et si [σ(b)≠b et σ(α(b))=α(b)] ou [σ(b)=b et σ(α(b))≠α(b)].

☛ *arête isolée:*

Une arête a= {b, α(b)} est dite isolée si elle est pendante et si σ(b)=b et σ(α(b))=α(b).

☛ *arête de bord:*

une arête est dite de bord si elle n'est pas pendante.

III-2.2 Cartes Géométriques

Les objets définis précédemment sont de nature purement topologique. Dans la suite on supposera travailler dans un plan euclidien et la carte sera représentée dans le plan.

Soient une carte $C=(B,\alpha,\sigma)$ et une représentation $\Re=(\pi, \lambda)$ où:

- π une application de l'ensemble des brins B dans R^2 qui fait correspondre à un brin b un point de R^2 qui vérifie:

$\pi(b)=\pi(b') \Leftrightarrow b \in \sigma^*(b)$

On appellera sommets géométriques de la carte géométrique $C=(B,\alpha,\sigma,\Re)$ les points p de R^2 tels qu'il existe $b \in B$ avec $\pi(b)=p$.

- λ est une application qui fait correspondre à un brin b de B une ligne polygonale orientée $l=(p_0, p_1,.....,p_n)$ avec $n>0$ *et* $\forall\ i,\ 0 \le i \le n,\ p_i \neq p_{i+1}$, qui est ouverte à son extrémité initiale $\pi(b)=p_0$ et ouverte à son extrémité finale $\pi(\alpha(b))=p_n$ et qui vérifie:

(i) $\forall\ i$ et j avec $0 \le i < j < n$, $p_ip_{i+1} \cap p_jp_{j+1}$est soit vide, soit réduite à un seul point.
(ii) $\forall\ b' \in B$ avec $b' \in \alpha^*(b)$, $\lambda(b)$ et $\lambda(b')$ sont disjoints.
(iii) $\forall\ b \in B$, $\lambda(\alpha(b))=(p_n,p_{n-1},.....,p_0)$.

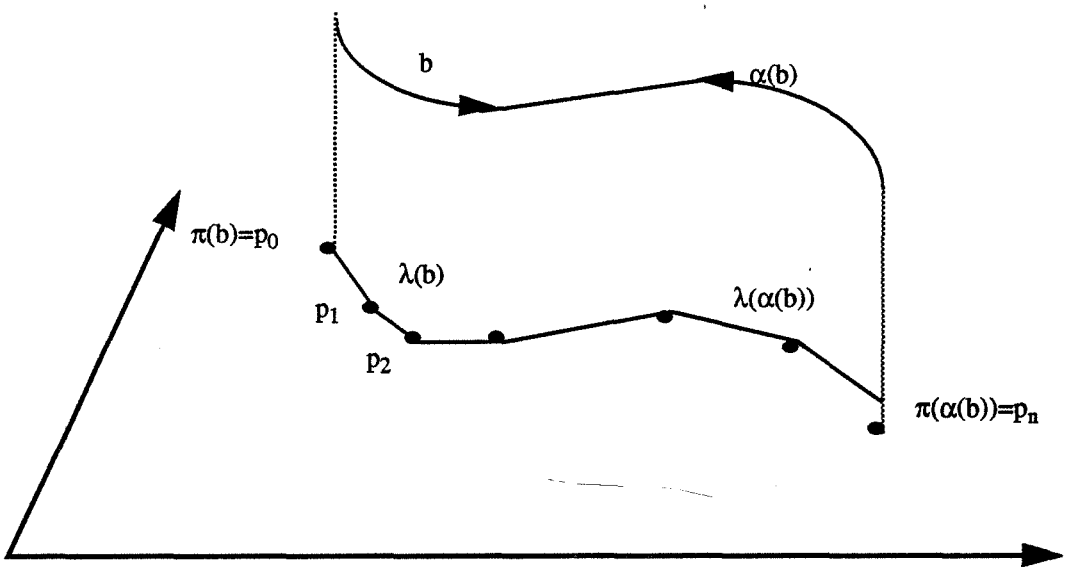


Figure23: Carte géométrique

III-2.2.1 Intérieur et extérieur d'un bord

On dit qu'un point p est *intérieur* à une ligne polygonale fermée $(p_0, p_1,.....,p_n)$, si:

$$\left(\forall i \le n, p \ne p_i, \sum_{i=0}^{n} angle\left(\overline{pp}_i, \overline{pp}_{i+1}\right)\right) = 2\pi$$

On dit que la ligne polygonale l_1 est *intérieure* à une ligne polygonale orientée fermée l_2 si tous les points de l_1 sont *intérieurs* à l_2.

Etant donné un ensemble $\wp$ de lignes polygonales du plan et une ligne polygonale orientée fermée $l \in \wp$, soit $\{l_1,l_2,....,l_k\} \subset \wp$ l'ensemble des polygones *intérieurs* à l. Un point p est dit *directement intérieur* à l relativement à $\wp$ si:

(i) p est intérieur à l;
(ii) $p \notin l_1 \cup l_2 ..... \cup l_k$;
(iii) $\forall i \in \{1,2,...k\}$ p n'est pas intérieur à l_i

Etant donné le même ensemble $\wp$ et l_1 et l_2 appartenant à $\wp$, on dit que l_1 est *directement intérieure* à l_2 relativement à $\wp$ si l_1 est *intérieure* à l_2 relativement à $\wp$ et s'il n'existe pas une ligne l de $\wp$ telle que l soit *intérieure* à l_2 et l_1 soit *intérieure* à l relativement à $\wp$.

Si l'on considère la figure 24 où $\wp = \{l_1, l_2, l_3\}$, les lignes l_2 et l_3 sont *intérieures* à l_1, l_2 est *directement intérieure* à l_1 et l_3 est *directement intérieure* à l_2 mais pas à l_1. De plus, la ligne l_3 n'a pas de points *intérieurs*.

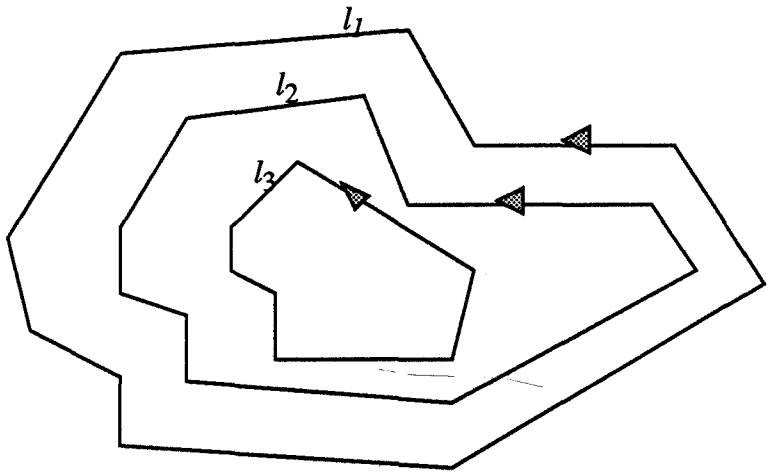


Figure 24: inclusion des bords

III-2.2.2 Arbre d'inclusion d'une carte géométrique

Soit une carte planaire $C = (B, \alpha, \sigma, \Re)$. On peut représenter les relations d'inclusion de la carte C à l'aide d'un arbre binaire tel que celui représenté sur la figure 25. Cet arbre est appelé arbre d'inclusion de la carte; il est construit de la manière suivante:

• Sa racine est un bord virtuel infini, contenant tous les bords de la carte. Ce bord est par définition interne.

• Un bord interne est forcément contenu dans un bord externe (sauf le bord infini), et inversement, un bord externe est forcément contenu dans un bord interne.

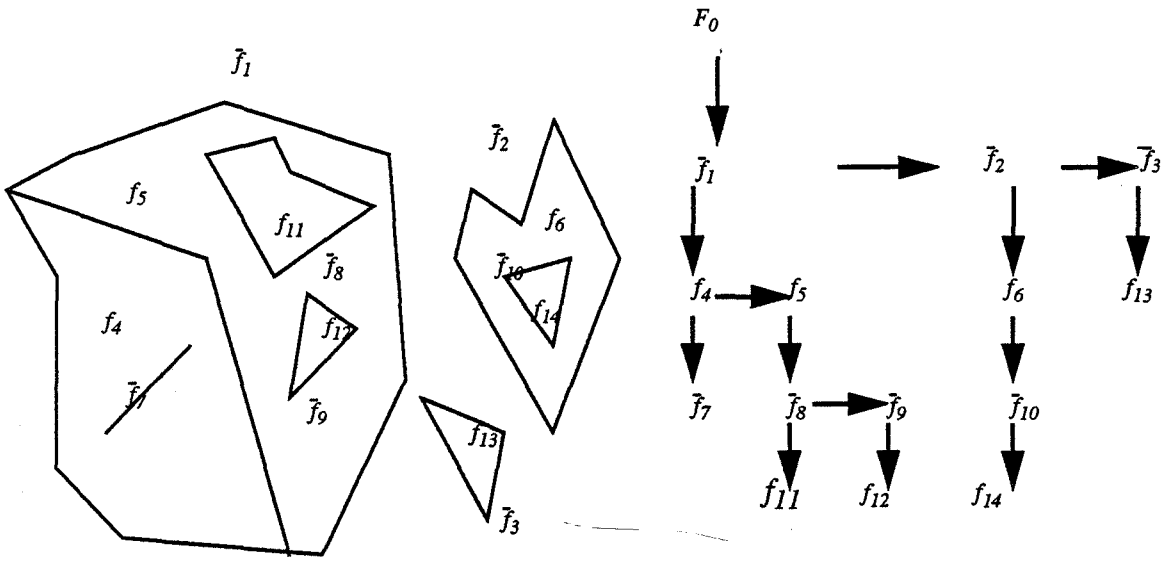


Figure 25: Arbre d'inclusion d'une carte géométrique

F_0 est la racine;
$\bar{f}_1, \bar{f}_2, \bar{f}_3, \bar{f}_7, \bar{f}_8, \bar{f}_9, \bar{f}_{10}$ sont des bords externes
$f_4, f_5, f_6, f_{11}, f_{12}, f_{13}, f_{14}$ sont des bords internes

III-3 CONSTRUCTION NON INCREMENTALE D'UNE CARTE PLANAIRE

III-3.1 Initialisation de la carte planaire

La construction de la carte planaire se fait en trois étapes: dans un premier temps on construit une structure de données locale pour les sommets et les arêtes. Au début, la structure de données n'est pas cohérente, car elle ignore les intersections entre les arêtes. Ainsi, dans un deuxième temps, les points d'intersection entre les arêtes sont-ils déterminés (construction de la CP-locale). Notons toutefois que les sommets ayant les mêmes coordonnées et les arêtes superposées seront confondus.

Un ordre lexicographique est défini sur les sommets: l'x-ordre est défini sur les sommets en les ordonnant par abscisse croissante. Pour deux sommets ayant même abscisse, le plus petit est celui de plus petite ordonnée. Cet ordre est total.

Soit une arête $a=(s_1,s_2)$ définie par ses sommets; l'arête a "naisse" en s_1 si est seulement si *x-ordre*(s_1) <*x-ordre*(s_2).

Une arête est aussi décrite par deux brins. Le premier brin a pour sommet le

premier sommet de l'arête; le deuxième brin a pour sommet le deuxième sommet de l'arête. Cette distinction entre brin et sommet est nécessaire car il est possible de définir sur les brins d'un sommet un ordre total: tout brin fait avec la verticale descendante du sommet un angle $\alpha \in]0,2\pi]$. Les brins sont ordonnés suivant leur angle. Cet ordre est appelé α-*ordre*.

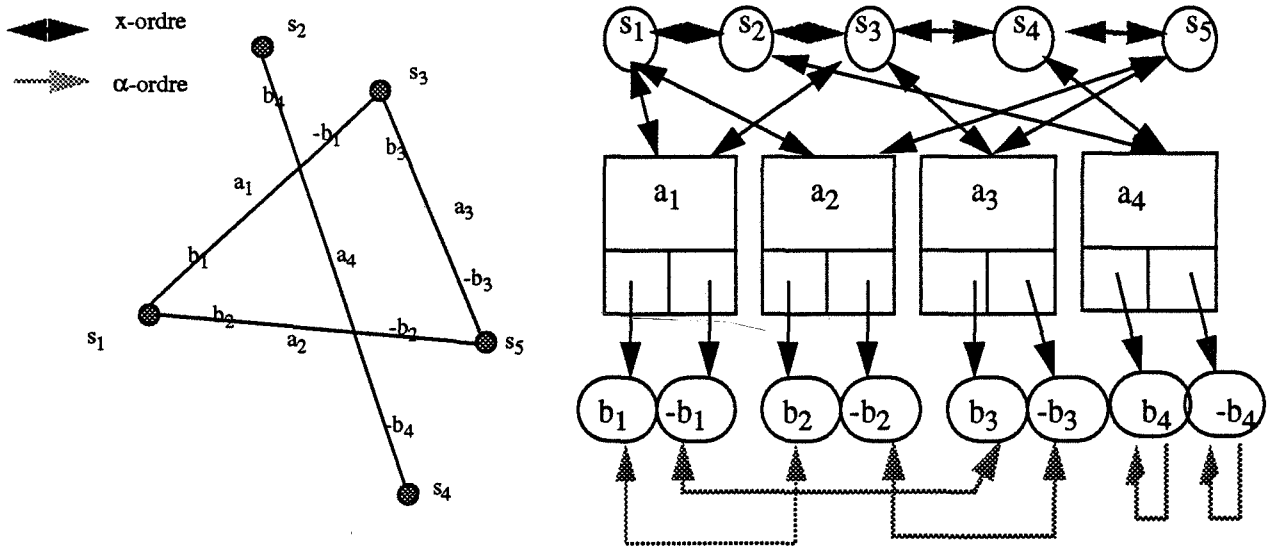


Figure 26: structure de données construite par l'étape d'initialisation

Durant la construction de la carte planaire, nous avons convenu de confondre les sommets qui sont à l'intérieur d'un cercle de rayon fixé; ainsi, un sommet n'est créé qu'une seule fois. Sur chaque brin est noté le brin suivant selon l'α-ordre, ce brin est appelé brin adjacent; de plus le dernier brin d'un sommet pointe vers le premier brin du même sommet. L'α-ordre est donc défini par une liste bouclée de brins. Le nombre d'arêtes incidentes en un sommet n'est bien sûr pas limité, mais fini.

III-3.2 Construction de la CP_locale

Etant données les deux notions (*x_ordre*, α-*ordre*), la construction de la CP_locale consiste à calculer les points d'intersection des arêtes initialement connues et à faire la mise à jour de la structure de données en insérant les nouveaux sommets associés aux points d'intersection et en remplaçant chaque arête contenant un point d'intersection par deux arêtes. L'insertion se fait en maintenant l'*x-ordre* des sommets et l'α-*ordre* des brins autour de leur sommet.

III-3.2.1 Recherche des points d'intersection

Pour trouver les points d'intersection entre N arêtes, la méthode la plus simple

consiste à tester l'intersection entre toutes les paires d'arêtes. Cette méthode n'est pas performante car elle a une complexité en $O(N^2)$. Nous nous sommes donc intéressés à l'algorithme de Bentley-Ottman [Bent79] qui offre de meilleures performances théoriques.

La méthode de Bentley_Ottman exploite et gère deux ordre: l'*x-ordre*, déjà défini, et l'*y-ordre*. L'*y-ordre* est défini sur un sous ensemble des arêtes, dites "actives", qui coupent une droite parallèle à l'axe des y; cette droite est appelée "droite de balayage". L'*y-ordre* classe les arêtes actives sur l'ordonnée croissante de leur point d'intersection avec la droite de balayage; ces points sont tous distincts pour les droites de balayage ne passant ni par un sommet, ni par un point d'intersection. Bien sûr, l'ensemble des arêtes actives, et l'*y-ordre*, dépendent de la position de la droite de balayage.

Le principe de cet algorithme, qui utilise une droite de balayage verticale se déplaçant de la gauche vers la droite, se base essentiellement sur deux remarques: premièrement, les seules droites de balayage utiles sont celles qui passent par un sommet ou par un point d'intersection déjà connu. Les modifications de l'*y-ordre* se font en éliminant les arêtes mortes et en insérant les arêtes naissantes, qui deviennent actives. Deuxièmement, deux arêtes ne peuvent se couper que si et seulement si elles sont contiguës selon l'*y-ordre* pour une droite de balayage donnée.

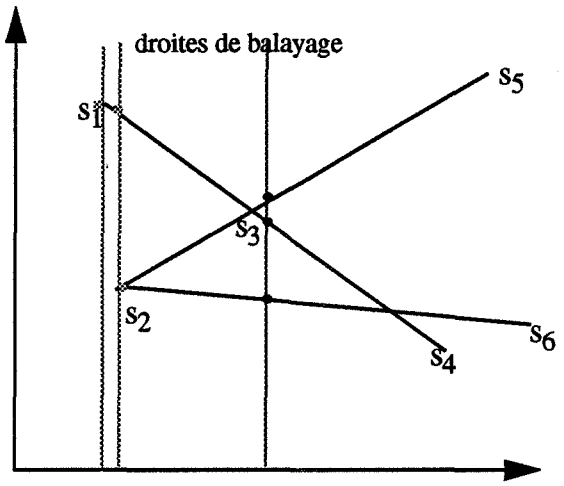


Figure 27: x-ordre, y-ordre et droites de balayage

L'algorithme peut être décrit de façon informelle de la manière suivante:

- La droite de balayage est initialement vide, puis elle est déplacée de la gauche vers la droite; elle balaie successivement tous les sommets et tous les points d'intersection, qui sont trouvés au fur et à mesure.

- Au cours du balayage, l'*y-ordre* est généré ainsi: en chaque sommet ou point d'intersection, les arêtes mortes sont enlevées et les arêtes naissantes sont

insérées en bonne place. Lors de cette modification, certaines arêtes deviennent contiguës dans l'*y-ordre*; cette adjacence entre deux arêtes est une condition nécessaire pour qu'elles s'intersectent; il suffit pour chaque nouvelle adjacence, de tester les arêtes contiguës.

✍ Si deux arêtes deviennent adjacentes et convergent vers la gauche, alors elles ont peut-être un point commun, mais il est déjà connu. Ces deux arêtes ont déjà été adjacentes dans l'y-ordre, il est donc inutile de tester leur éventuelle intersection.

✍ Si deux arêtes deviennent adjacentes et convergent vers la droite, et si leurs sommets de naissance ne sont pas identiques, alors on a trouvé peut-être un point d'intersection. Si tel est le cas, on crée un nouveau sommet et on remplace chacune des deux arêtes par deux nouvelles arêtes. Le point d'intersection devient un sommet comme les autres et il est inséré dans l'x-ordre.

✍ Etablir la liste des cas où deux arêtes deviennent adjacentes pour un sommet S n'est pas difficile:

- si S est un sommet de mort (voir la figure 28c), alors l'arête immédiatement au dessous (dite active basse) et l'arête immédiatement au-dessus (dite active haute) deviennent adjacentes.

-si S est un point de vie ou de naissance (voir les figures 28a, 28b), alors l'arête naissante la plus basse et l'active basse et l'arête naissante la plus haute et l'active haute, deviennent adjacentes.

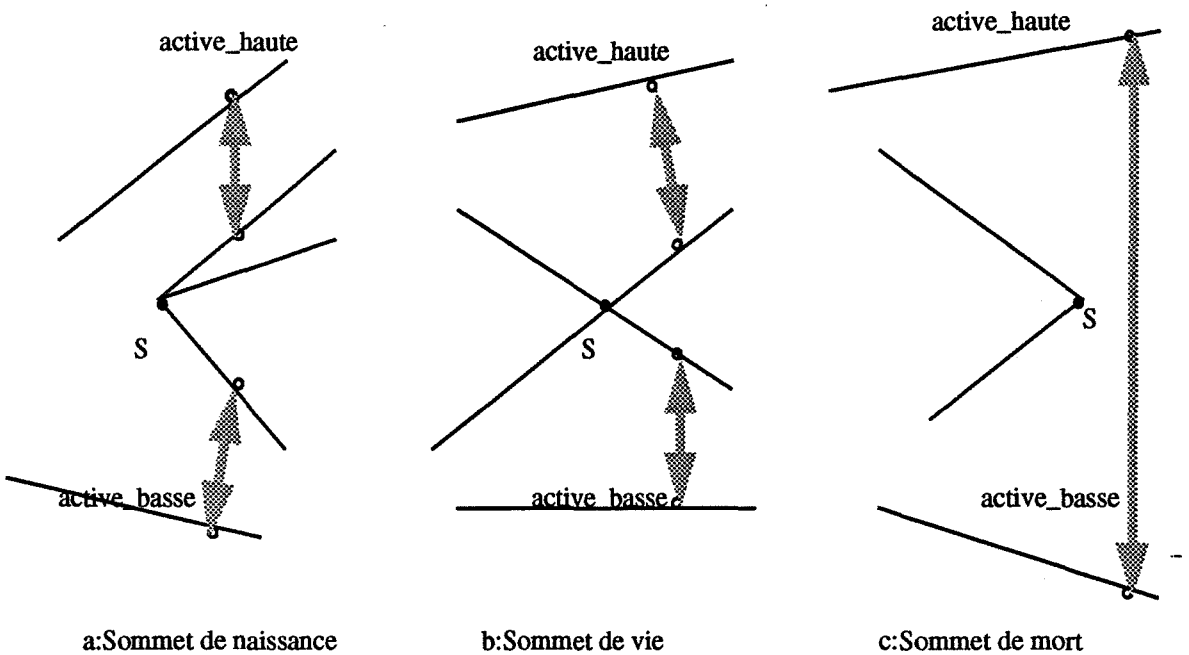


Figure 28: adjacences nouvelles selon l'y-ordre

Nous pouvons maintenant formuler l'algorithme de Bentley-Ottman de façon plus précise.

```
BENTLEY_OTTMAN
  {
    insérer dans x-ordre tous les sommets connus, ordonnés;
    initialiser l'y-ordre comme vide;
    pour chaque sommet S selon l'x-ordre, faire
    {
    si S est:
          - un sommet de mort:
                enlever les arêtes mortes de l'y-ordre;
                comparer(active_haute, active_basse);
          -un sommet de naissance:
                insérer les arêtes naissantes dans l'y-ordre;
                comparer(la plus basse des nouvelles-nées, active_basse);
                comparer(la plus haute des nouvelles-nées, active_haute);
          - un sommet de vie:
                enlever les arêtes mortes de l'y-ordre;
                insérer les arêtes naissantes dans l'y-ordre;
                comparer(la plus basse des nouvelles-nées, active_basse);
                comparer(la plus haute des nouvelles-nées, active_haute);
    }
  }


    comparer (a,b:arête)
    {
    si (a et b existent) et (a et b convergent vers la droite) et
      (leurs sommets droits sont distincts) et (a et b se recouvrent en y)
    alors
          {
          i ←point d'intersection de a et b;
          si (i existe)
                alors
                      {
                mettre à jour les structures de données de CP;
                insérer i dans l'x-ordre;
                }
          }
    }
```

Pour analyser la complexité de l'algorithme de Bentley_Ottman, appellons N le nombre de segments initiaux et K le nombre d'intersections; dans le pire des cas,

K=N(N-1)/2=O(N^2), quand tous les segments s'intersectent; dans ce cas, la complexité de l'algorithme est en O(N^2) qui est égale à celle de l'algorithme naïf. Si K n'est pas en O(N^2), la complexité de l'algorithme de Bentley-Ottman est en O((N+K)log(N)).

III-3.2.2 La précision

Il faut cependant noter que les difficultés de la méthode de Bentley-Ottman ne proviennent pas de l'algorithme, mais des imprécisions numériques inhérentes à l'utilisation des nombres flottants. En effet, contrairement à d'autres algorithmes où les erreurs restent locales, dans Bentley-Ottman il y a propagation de ces erreurs. Michellucci [Mich87] a étudié l'effet de l'imprécision numérique sur les méthodes de calcul des points d'intersection de deux segments. Il a montré que les conséquences sont bien plus graves avec l'algorithme de Bentley-Ottman.

L'exemple suivant, présenté par Gangnet [Gang87], illustre les effets néfastes de l'imprécision numérique sur la méthode de Bentley-Ottman. Par commodité, on suppose qu'une arithmétique flottante en base 10 avec P=4 chiffres significatifs est employée. On notera x* la meilleure valeur flottante représentant le réel x.

Soit N un entier >0;

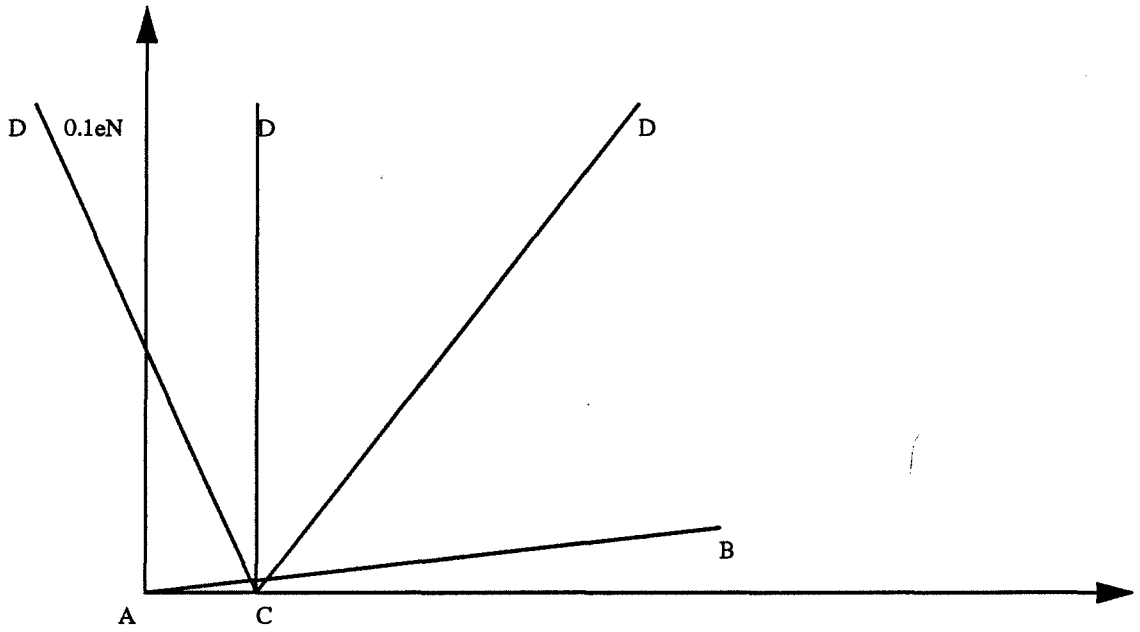


Figure 29: configuration de GANGNET

On définit les points A, B, C, D_u par:
A=(0.0e0, 0.0e0) B=(0.1eN, 0.1e1);
C=(0.0e1,0.0e0) D_u=(0.1e1+u, 0.1eN) où u est un paramètre
Les droites (AB) et (CD_u) ont pour équations:

(AB): (0.1e1)x + (0.1eN)y=0
(CD_u): (0.1eN)x + (u)y + 0.1eN=0

Δ= (0.1e(2N-1) -(0.1e1)u
Δ*=0.1e(2N-1) est la valeur flottante la plus proche de Δ, pour |u|<0.0001e(2N-1)

calculons maintenant les points d'intersection I_u des droites (AB) et (CD_u)
Δx= 0.1e(2N-1)= Δx*
Δy= 0.1eN= Δy*
x= [0.1e(2N-1)]/[0.1e(2N-1)-(0.1e1u)]
y=(0.1eN)/[0.1e(2N-1)-(0.1e1u)]
x*=[0.1e(2N-1)]/[0.1e(2N-1)]
y*=(0.1eN)/[0.1e(2N-1)]
d'ou I*=(0.1e1,0.1e-N+2)

Ainsi pour toutes les valeurs de u telles que |u|<0.0001e(2N-1), les droites (CD_u) ont le même point d'intersection avec la droite (AB), ce qui est en contradiction avec les propriétés du plan euclidien.

Etudions maintenant les conséquences de ce qui précède sur l'algorithme de Bentley-Ottman:

soit N=2 et |u|<0.1;
A=(0,0) B=(10,1) C=(1,0) D_u=(1+u,10) donc D_u est compris entre (0.9,10) et (1.1,10)

I_u* est le point d'intersection des droites (D_uC) et (AB) et D_u<I_u*<C; or, l'algorithme ne connaît les coordonnées que par leurs représentations flottantes: selon ce test, D_u<C<I_u*, donc I_u* n'appartient pas à la droite (D_uC). Ainsi donc, la méthode de Bentley-Ottman oublie un point d'intersection, ce qui perturbera la construction de la carte planaire.

Les imprécisions numériques ont des effets graves, non seulement sur le calcul des points d'intersection, mais aussi sur l'orientation des brins autour des sommets. En effet, l'orientation des brins autour des sommets est fondée sur une comparaison d'angles entre deux arêtes ayant un sommet en commun. L'incertitude angulaire devient importante quand la longueur de l'arête est petite.

En effet, les choses se compliquent lorsqu'on veut comparer deux nombres réels. Ceci peut être mis en évidence en comparant, sur une seule machine les expansions binaires de U=7.8, et V= 78.0 /10.0 (bien sûr, U=V et U-V=0.0)

U= 111.11001100110011001100110011001100110011001100110011001100110011001
V= 111.11001100110011001100110011001100110011001100110011001100110011101
U-V=0.00000000000000000000000000000000000000000000000000000000000000001

Le test d'égalité entre U et V prend la valeur FAUX sur la machine, alors qu'une demande d'affichage de la différence entre u et v donnera 0.0: la tolérance d'erreur n'est pas la même pour une opération d'affichage et un test logique.

Les effets de l'imprécision numérique peuvent provoquer des erreurs fatales, qui font que la CP_locale est incohérente avec la réalité. Différentes approches ont été proposées pour résoudre ce type de problème:

- associer des valeurs de tolérance aux valeurs de départ et assurer la cohérence des algorithmes en maintenant les valeurs de tolérance à chaque opération [Sega90]. Ce type d'approche ralentit de façon appréciable l'algorithme puisque on doit effectuer des retours en arrière pour assurer la transitivité de la relation de coïncidence entre deux points.

- utiliser l'arithmétique rationnelle [Mich87]. Un rationnel est composé d'un numérateur entier et d'un dénominateur entier non nul, dont on suppose qu'il sont premiers entre eux. Un tel nombre est représenté par une structure de données composée d'un signe (sur un bit si cela a un sens sur la machine hôte) et de deux listes chainées de chiffres écrits dans une base B choisie. Les calculs se font alors sans erreurs et il n'y a pas d'approximations à gérer. Il faut implémenter les opérations rationnelles et les coûts de calcul ne sont pas forcément négligeables.

- ne calculer qu'en fonction des valeurs numériques de départ et utiliser des raisonnements symboliques pour résoudre les ambiguïtés [Hoff89].

- utiliser un algorithme mixte [More90], où l'on dispose de deux évaluateurs: une arithmétique à précision finie livrée avec la machine hôte et une arithmétique exacte (arithmétique rationnelle). L'algorithme consiste alors à calculer d'abord en précision finie, puis, uniquement quand on a des doutes, c'est à dire à l'intérieur d'un seuil de "réticence" prédéfini, on prolonge les résultats déjà obtenus par des calculs en arithmétique rationnelle.

- Verroust [Verr90] utilise le même raisonnement que Segal [Sega90], en associant des valeurs de tolérance aux sommets et aux arêtes, mais elle introduit des "sauts" entre certains sommets au lieu d'augmenter systématiquement les tailles des valeurs de tolérance. Ces sauts concernent des points qui ne peuvent être distants entre eux de plus de ε (valeur de tolérance) et qui ne sont portés par aucune arête. Ces sauts permettent de localiser plus finement les lieux de croisement entre deux arêtes que dans l'approche de Segal.

III-3.3 Construction de la CP_globale

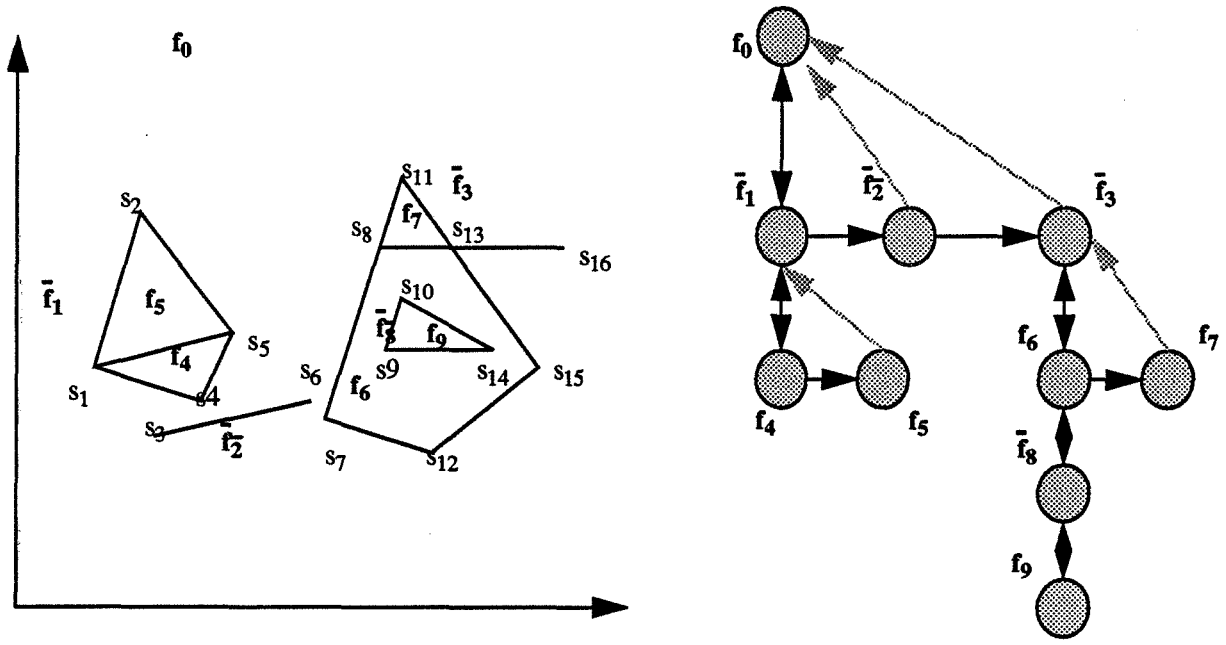
A ce stade, tous les points d'intersection ont été déterminés. L'algorithme doit créer les bords, les insérer dans l'arbre d'inclusion et marquer sur chaque brin, le bord

qui l'emprunte. Pour déterminer un bord, il suffit de pouvoir retrouver un brin lui appartenant; la CP_locale permet le parcours. A tout bord, on attache donc son premier brin: c'est celui qui a le sommet le plus petit selon l'x-ordre. Ce sommet est le sommet de naissance du bord. Mais plusieurs brins peuvent y naître; dans ce cas, le brin le plus petit selon l'α-ordre sera le brin attaché au bord.

Il faut aussi représenter l'ordre naturel induit par les inclusions entre les bords. L'inclusion entre les bords est représentée par un arbre généalogique binaire; chaque bord pointe sur son bord père, son bord fils et son bord frère cadet. Entre les bords frères est défini un ordre total qui est celui de leur sommet de naissance; dans le cas où deux bords ont le même sommet de naissance, ils seront ordonnés selon l'α-ordre de leur brin de naissance.

Un bord externe est obtenu en parcourant la CP_locale dans le sens des aiguilles d'une montre, alors qu'un bord interne est obtenu en faisant le parcours inverse.

On a introduit une donnée supplémentaire, appelée arête de butée d'un sommet, qui sert à localiser rapidement une arête ou un sommet. Dans le cas d'un sommet de naissance, la butée est la première arête active basse ne naissant pas en ce sommet lors du passage de la droite de balayage. Dans les autres cas, (vie ou mort), la butée est la première arête, selon l'α-ordre, mourant en ce sommet (voir la figure 30).



s_{15} a pour butée l'arête $[s_{11},s_{15}]$

s_9 a pour butée l'arête $[s_7,s_{12}]$

$\bar{f}_i$: bord externe; f_i : bord interne

f_0 est un bord interne virtuel

Figure 30: La CP_locale et son arbre de bords

Le pseudocode suivant décrit l'algorithme de construction de la CP_globale à partir de la CP_locale:

```
Construction de la CP_globale {
        pour chaque sommet S, pris dans l'x-ordre, faire
            {
            pour chaque arête naissante en S, selon l'α-ordre faire
                si (le bord du brin 2 de l'arête n'existe pas encore)
                 alors{
                    créer bord: B;
                    parcourir les brins de B en les marquant comme empruntés par B;
                    affecter le brin de naissance de B;
                    si (S est une naissance) et (l'arête est la première incidente de S)
                     alors{
                            B est externe;
                            si (la butée de S n'existe pas)
                                alors B' ← bord virtuel de la carte
                                sinon B' ← bord du brin 1 de la butée de S;
                            si (B' est interne)
                            alors le père de B est B'
                            sinon le père de B est le père de B'
                        }
                     sinon {
                    B est interne;
                    B' ← bord de brin 2 de l'arête de naissance de B
                    si (B' est externe)
                            alors le père de B est B';
                    sinon le père de B est le père de B'
                    }
            ajouter B à la fin de la liste des fils du père de B;
            }
        }
}
```

La construction exposée ci-dessus d'une carte planaire pose des problèmes dès lors que l'on se trouve dans un environnement interactif. Bien sûr, il est toujours possible de recalculer toute la carte, mais alors il n'est plus possible (du moins facilement) d'associer des attributs aux différents objets de la carte. D'où l'idée d'introduire la construction incrémentale des cartes planaires.

III-4 CONSTRUCTION INCREMENTALE D'UNE CARTE PLANAIRE

Comme le logiciel développé est une interface graphique, l'un des objectifs de ce travail est l'interactivité. En particulier, il faut pouvoir insérer des murs d'un bâtiment, des câbles d'un réseau. Ceci suppose que l'on puisse effectuer un calcul dynamique. La construction incrémentale d'une carte planaire consiste en l'insertion et la suppression incrémentale de segments, ainsi que le calcul dynamique des nouveaux points

d'intersection et la mise à jour des structures de données. L'algorithme de Bentley-Ottman n'est pas applicable, du moins tel qu'il a été décrit, pour la construction de la carte planaire locale. Pour la construction de la carte planaire globale, Gangnet [Gang89] propose une méthode de mise à jour de l'arbre d'inclusion des bords basée sur la classification des arêtes ajoutées ou supprimées. D'autre solutions peuvent être considérées qui utilisent des structures de données différentes des cartes planaires.

Nous proposons ici une solution qui utilise l'algorithme de Bentley-Ottman amélioré pour la construction de la CP_locale. Pour la construction de la CP_globale à partir de la CP_locale (cet phase est très rapide relativement à l'algorithme de Bentley_Ottman) une solution brutale a été implémentée: elle consiste à reconstruire l'arbre d'inclusion et à le comparer à l'ancien arbre.

III-4.1 Mise à jour de la CP_locale

Considérons d'abord la suppression d'une ou de plusieurs arêtes: la suppression des brins dans la CP_locale est triviale; la principale difficulté est posée par la mise à jour des butées; certaines arêtes supprimées peuvent en effet être les butées de certains sommets de naissance. L'ajout de nouvelles arêtes est encore plus difficile: non seulement des butées peuvent être modifiées, mais les nouveaux points d'intersection doivent être déterminés.

Supposons que nous ayons à faire l'insertion d'une arête sur une carte planaire déjà existante. Intuitivement, l'algorithme d'insertion est le suivant (voir la figure 31):

- les deux sommets, extrémités du segment, sont insérés dans la liste des sommets selon l'x-ordre.

- La droite de balayage est initialement vide, puis elle est déplacée de la gauche vers la droite; elle balaie successivement tous les sommets jusqu'au premier sommet de la nouvelle arête et fait la mise à jour de la pile des arêtes actives. Durant cette phase, il n'existe pas de nouveaux points d'intersection.

- A partir du premier sommet de la nouvelle arête, on applique l'algorithme de Bentley_Ottman qui permettra d'insérer les nouveaux sommets d'intersection et d'ordonner les brins autour des sommets. L'algorithme de Bentley_Ottman s'arrêtera quand le dernier sommet de la nouvelle arête sera atteint.

Considérons maintenant la suppression d'une arête d'une carte existante. Intuitivement l'algorithme de suppression est le suivant:

- supprimer les deux brins de l'arête et faire la mise à jour des sommets concernés.

- déplacer la droite de balayage de la gauche vers la droite; elle balaie successivement tous les sommets et fait la mise à jour des butées. Si un sommet n'a plus de brin, il est alors éliminé.

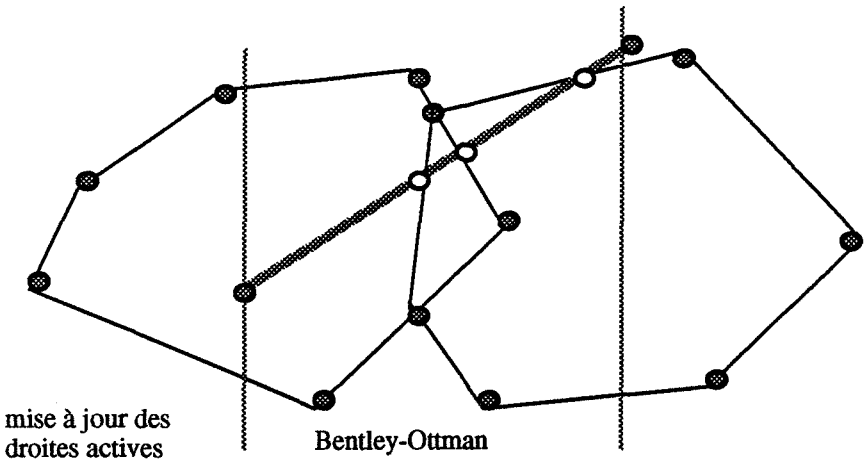


Figure 31 : Mise à jour de la CP_locale

III-4.2 Mise à jour de la CP_globale

III-4.2.1 Solution de Gangnet

La mise à jour de l'arbre d'inclusion des bords est une tache délicate. En effet, la suppression ou l'ajout d'une arête peut modifier complètement d'ordre d'inclusion des bords. Gangnet [Gang 89] a proposé un algorithme basé sur la classification des arêtes: selon le type de l'arête et de l'opération (ajout, suppression), il éclate ou fusionne les bords.

Gangnet classifie les arêtes comme suit (voir la figure 32):

*1- **terminale** dont les deux brins appartiennent au même bord interne.*
*2- **de connexion** dont les deux brins appartiennent au même bord interne.*
*3- **de bord** dont les deux brins appartiennent à deux bords internes distincts.*
*4- **de bord** dont un brin appartient à un bord interne et l'autre brin appartient à un bord externe.*
*5- **isolée**.*
*6- **terminale** dont les deux brins appartiennent au même bord externe.*
*7- **de connexion** dont les deux brins appartiennent au même bord externe.*

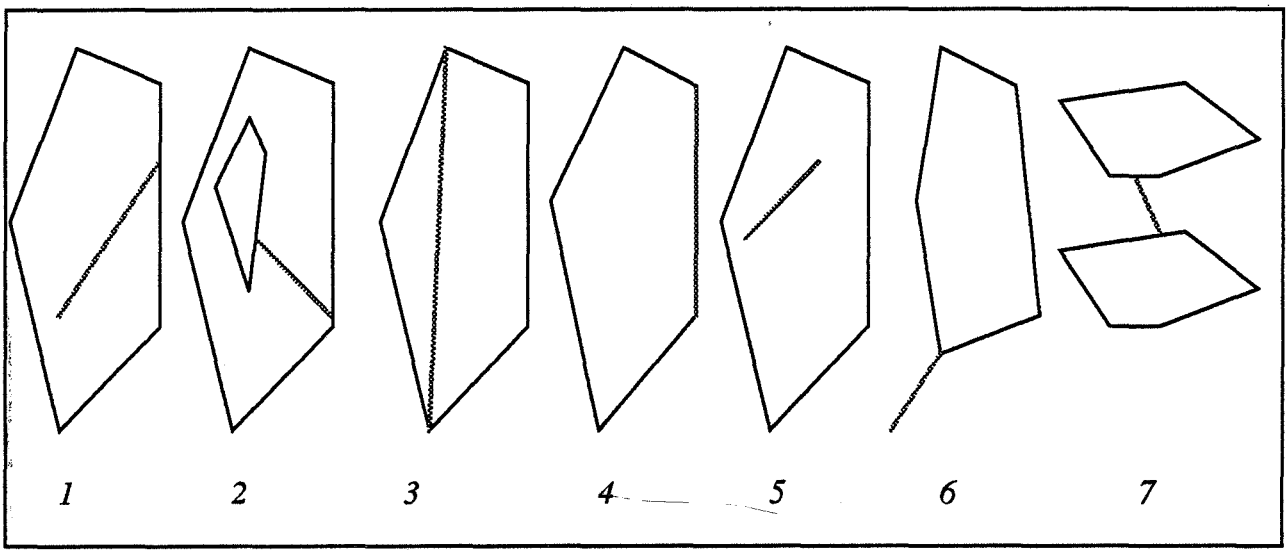


Figure 32 : Classification des arêtes par Gangnet

L'ajout d'une arête implique la création de ses brins et de ses sommets et la mise à jour de l'α-ordre. L'arbre d'inclusion est mis à jour après chaque modification en réalisant les actions suivantes (voir la figure 33):

cas du (type de l'arête)
{
cas 2:
un bord interne et un bord externe sont fusionnés en un seul bord interne.
cas 3:
un bord interne est éclaté en deux bords internes.
cas 4:
un bord externe est éclaté en un bord externe et un bord interne.
cas 5:
un bord externe est créé
cas 7:
deux bords externes sont fusionnés en un seul bord externe.
}

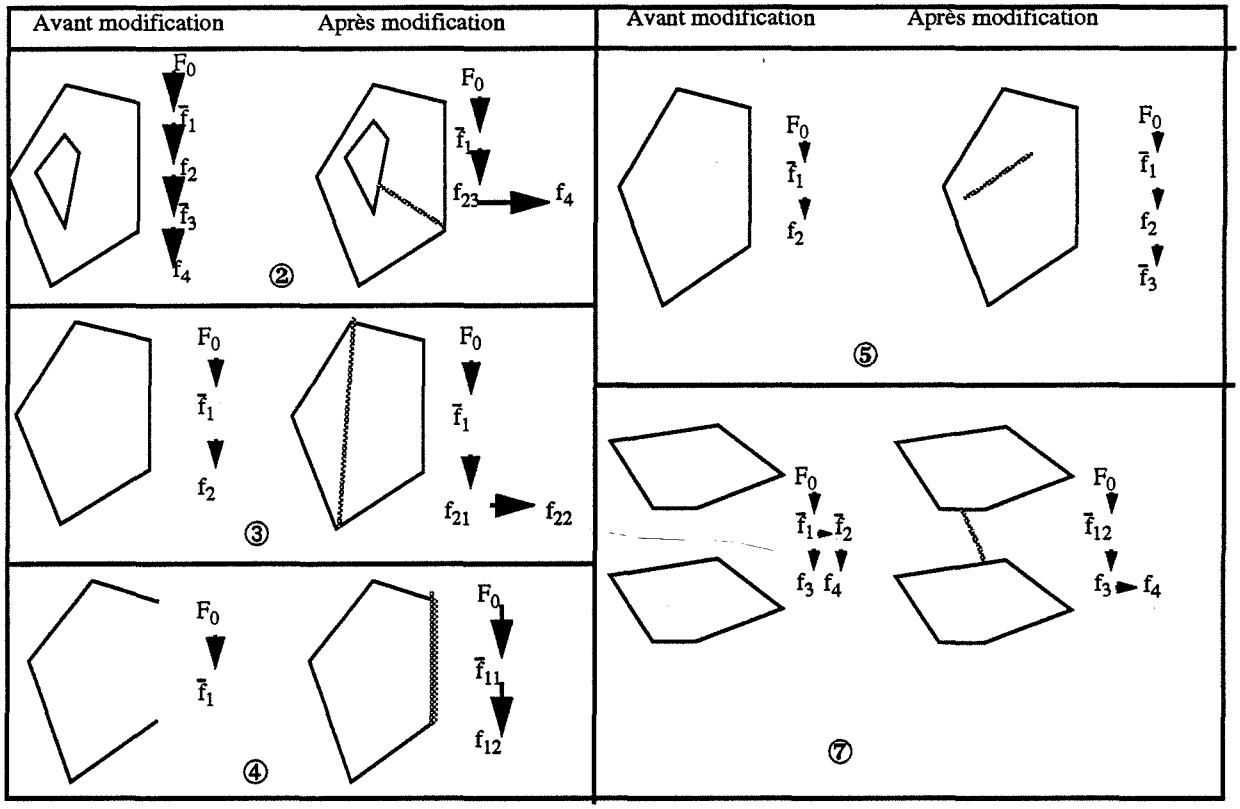


Figure 33 : Mise à jour de la CP_globale (ajout d'une arête)

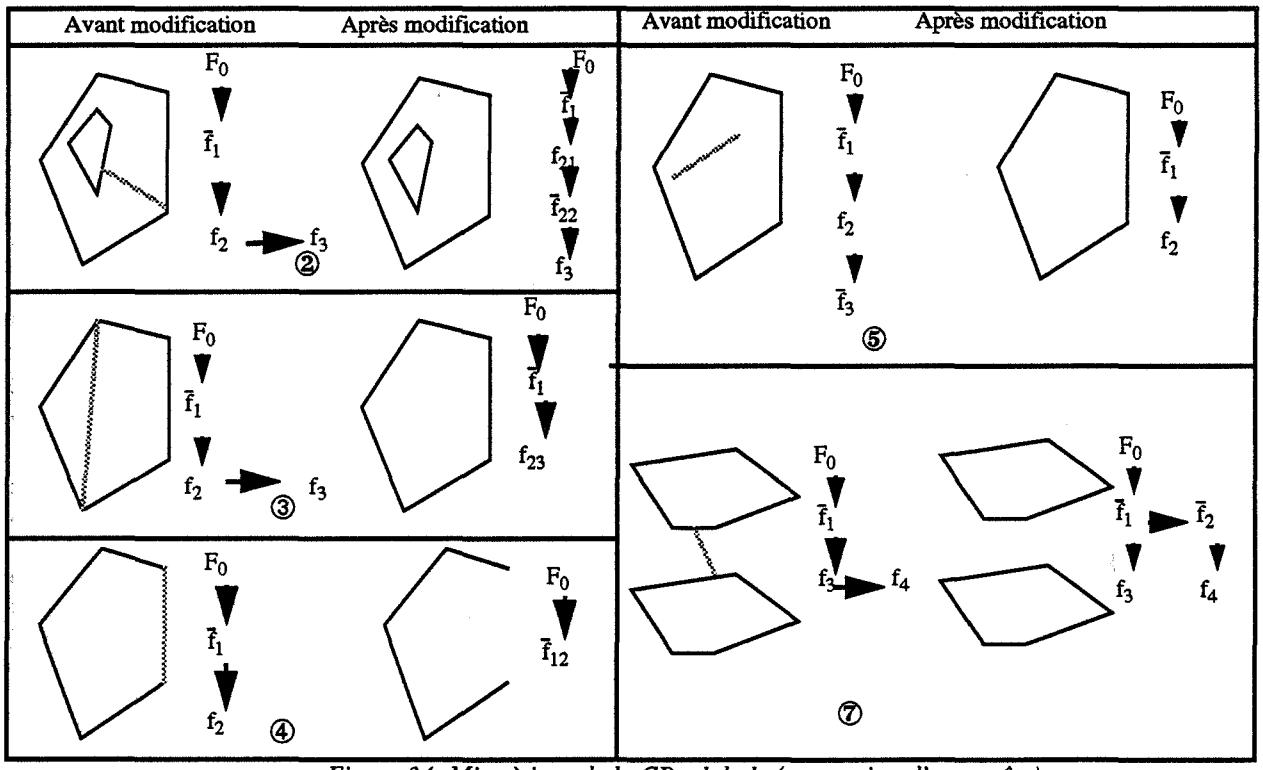


Figure 34: Mise à jour de la CP_globale (supression d'une arête)

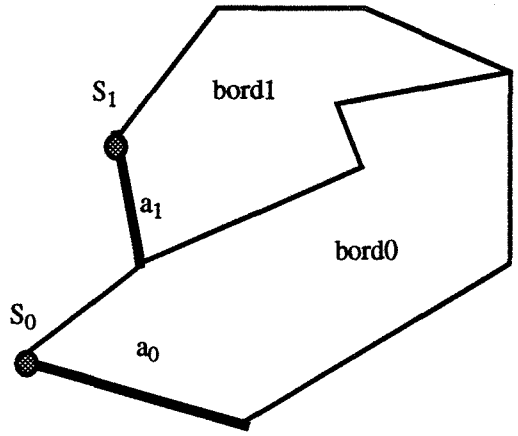
Quand une arête est supprimée, l'arbre d'inclusion est mis à jour avant toute modification de la CP_locale. Une fois que le type de l'arête est déterminé, la mise à jour de l'arbre d'inclusion se fait de la manière suivante (voir la figure 34):

cas du (type de l'arête)
{
cas 2:
un bord interne est éclaté en un bord interne et un bord externe.
cas 3:
deux bords internes sont fusionnés en un seul bord interne.
cas 4:
un bord externe et un bord interne sont fusionnés en un seul bord externe.
cas 5:
un bord externe est supprimé.
cas 7:
un bord externe est éclaté en deux bords externes.
}

La suppression et l'ajout d'une arête de type 1 ou de type 6 ne modifie pas l'arbre d'inclusion.

III-4.2.2 Notre solution

L'algorithme que nous proposons ici permet la reconstruction de l'arbre d'inclusion tout en gardant les informations qui lui sont associées. Cet algorithme se base sur la remarque suivante: tout bord de carte planaire est identifié par son sommet de départ (le plus petit sommet appartenant au bord selon l'x-odre) et par son arête de départ (la plus petite arête selon l'α-ordre autour du sommet de départ) (voir la figure 35).



bord0 est identifié par le sommet S_0 et l'arête a_0

bord1 est identifié par le sommet S_1 et l'arête a_1

Figure 35 : Identification des bords

Tenant compte de cette remarque, l'algorithme de reconstruction de l'arbre d'inclusion est le suivant (voir la figure 36):

```
pour toute modification de la carte planaire faire
{
mise à jour de la CP_locale;
sauvegarde de l'ancien arbre d'inclusion;
construction du nouvel arbre d'inclusion;
pour chaque bord du nouvel arbre d'inclusion faire
        {
        identifier le bord dans l'ancien arbre d'inclusion;
        si le bord existe alors
            {
            marquer le bord dans l'ancien arbre d'inclusion;
            transférer les attributs associés à ce bord de l'ancien arbre vers le
                                    nouveau;
            } finsi;
        } finpour;
détruire l'ancien arbre d'inclusion;
}
```

L'identification du bord dans l'ancien arbre d'inclusion consiste à chercher le bord qui à le même sommet de départ et la même arête de départ.

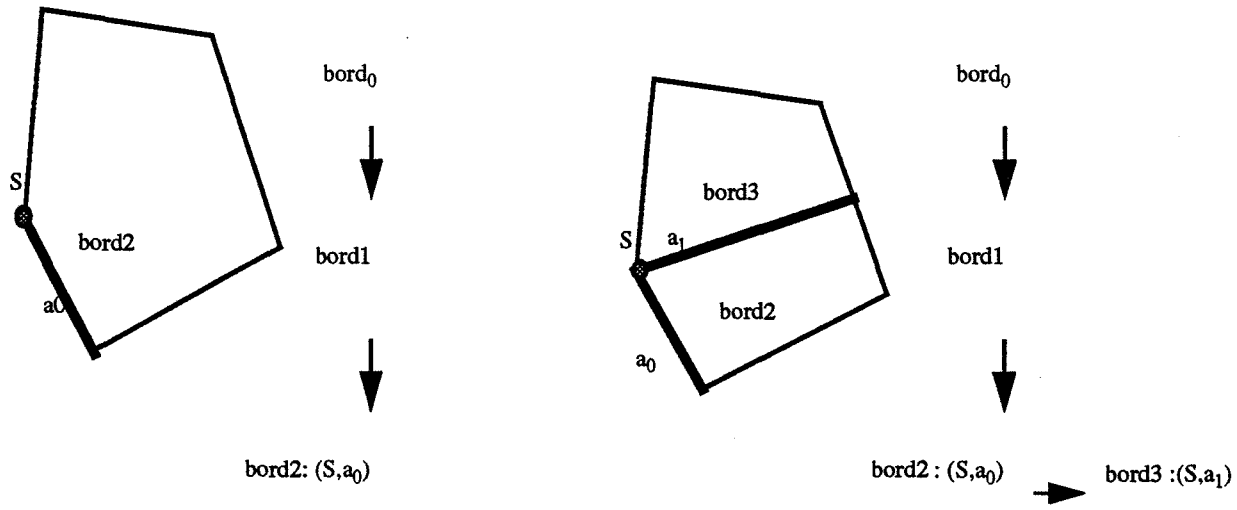


Figure 36 : Ajout de l'arête a_1 : on construit le nouvel arbre d'inclusion, on identifie le bord2 dans l'ancien arbre d'inclusion et on transfère ses attributs sur le nouvel arbre d'inclusion

III-5 CONCLUSION

Ce chapitre a présenté la structure de données de carte planaire, ainsi qu'un algorithme efficace de construction. Cependant les cartes planaires (ou des variantes) présentent quelques limitations notamment pour les opérations (rotation, transformation, inclusion,...) applicables aux cartes planaires.

Ce chapitre a montré aussi l'importance de l'effet des imprécisions numériques: l'imprécision de l'arithmétique flottante rend inutilisable l'algorithme de Bentley-Ottman; l'exploitation de la cohérence du plan est en effet incompatible avec l'imprécision numérique. Des méthodes moins efficaces résistent beaucoup mieux que l'algorithme de Bentley-Ottman aux imprécisions numériques.

L'arithmétique rationnelle permet de résoudre les problèmes d'imprécision numérique. Son coût doit cependant être pris en compte lors de l'évaluation des performances des algorithmes.

CHAPITRE IV

REALISATION

Ce chapitre présente les structures de données adoptées et les algorithmes importants pour la construction d'un outil graphique pour le système MMI2. La conception de l'outil de dessin est abordée selon des critères de flexibilité et de réutilisabilité de l'outil. On précisera notamment la dépendance de l'outil vis-à-vis de l'application et vis-à-vis du système de fenêtrage.

IV- REALISATION

IV- 1 INTRODUCTION

Le principal objectif du projet MMI2 est de développer une interface hautement interactive entre un utilisateur et une application de type système expert. Cette interface devrait permettre un dialogue multi-modal en mixant différents modes de communication tels que la langue naturelle, un langage de commande, le graphique et le gestuel. Qui plus est, l'interface doit faciliter un dialogue "coopératif" entre l'utilisateur et l'application et permettre ainsi à l'application, non seulement de poser des questions et de donner des réponses, mais aussi, d'élaborer et de justifier un raisonnement, de faire des suggestions et d'émettre des critiques.

L'application choisie est un système expert d'aide à la conception de réseaux informatiques. L'utilisateur peut saisir le plan du bâtiment et spécifier l'équipement matériel du réseau en manipulant des icônes. Il peut aussi demander à l'application d'analyser et de compléter le réseau fourni; le système engage alors un dialogue avec l'utilisateur en utilisant soit le mode graphique, soit la langue naturelle (voir la figure 37). Les réponses sont construites à partir des connaissances de l'application sur les réseaux et des connaissances du système sur l'utilisateur.

A travers les différents modes de communication, l'utilisateur exprime des actes de communication. Un acte de communication peut être soit une requête, soit une réponse. Chaque expert de mode (l'expert graphique, les experts de la langue naturelle, l'expert du langage de commande) construit une ou plusieurs expressions CMR exprimant l'acte de communication. Ces expressions sont envoyées au contrôleur de dialogue qui, après analyse, construit une réponse en interrogeant les différents experts du système. La réponse construite, le contrôleur de dialogue décide du mode de communication le plus approprié pour afficher la réponse à l'utilisateur.

Comme indiqué dans le chapitre 1, l'architecture de l'expert graphique présente deux niveaux (voir la figure 38): le premier est constitué d'outils graphiques interactifs permettant la visualisation et la manipulation de l'information. En particulier, ces outils permettent de visualiser les informations sous forme de bâtiments, de réseaux, d'histogrammes, de tables, de camemberts et de courbes. Des opérations graphiques telles que la sélection, le déplacement, la création et la destruction d'objets sont supportées par les outils graphiques. Ces opérations peuvent être exécutées par l'utilisateur ou par le gestionnaire graphique. Chacun de ces outils possède ses propres structures de données.

Lorsqu'on choisit le mode gestuel, les outils graphiques lui servent de support d'interaction. En effet, à travers le mode gestuel, l'utilisateur manipule des objets qui sont affichés dans les fenêtres des outils graphiques. En esquissant des gestes sur une fenêtre,

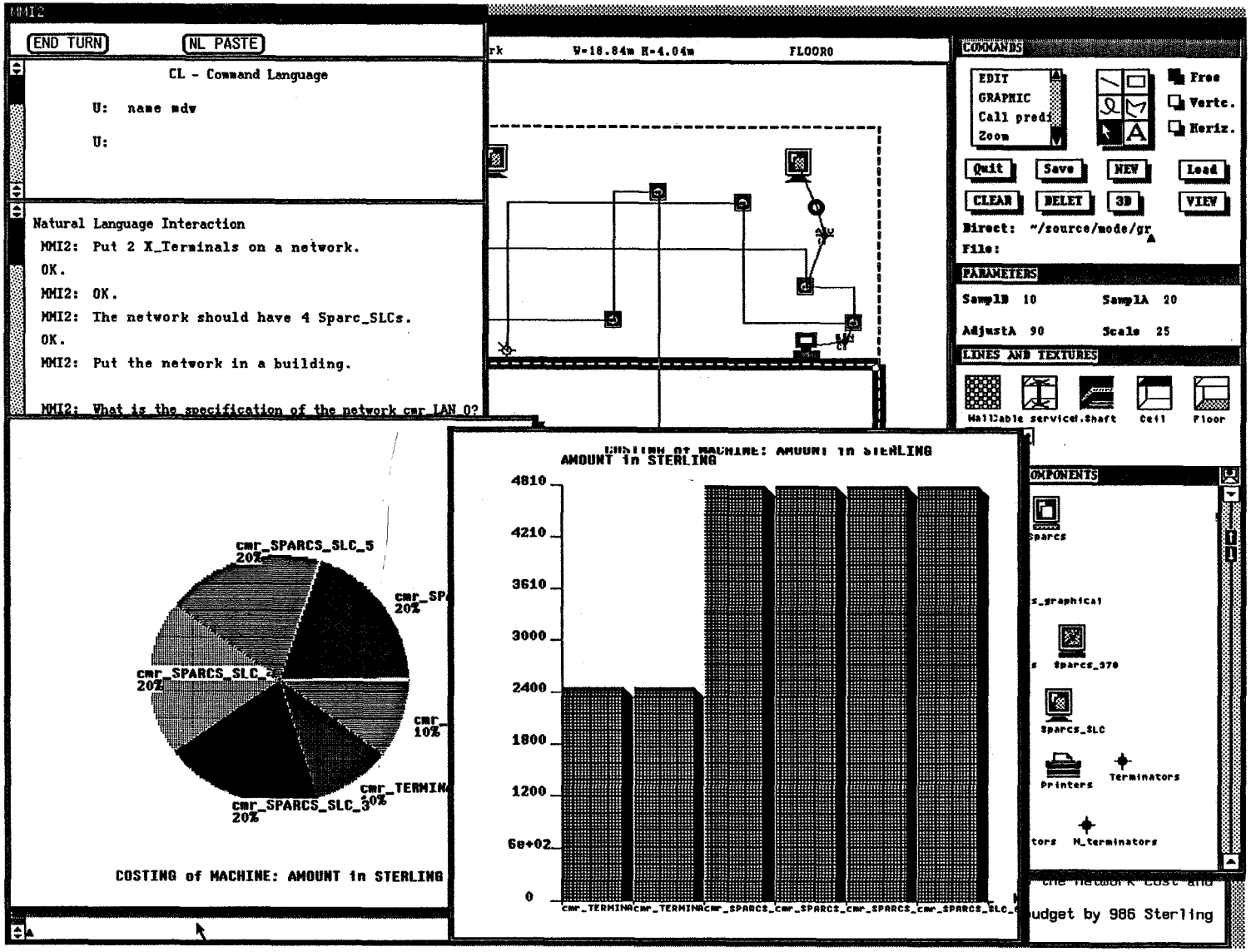


Figure 37:MMI2: une vue globale

les événements de la souris sont aiguillés vers le module gestuel; celui-ci reconnaît le symbole et demande à l'outil graphique d'exécuter la commande associée à ce "geste".

Le deuxième niveau est constitué du gestionnaire graphique qui assure la communication avec le reste du système. Le gestionnaire graphique traduit les expressions CMR en un ensemble d'actions permettant de piloter et de construire les structures de données de chaque outil graphique. En effet, quand le mode graphique est choisi, le gestionnaire graphique décide de l'outil graphique à utiliser pour visualiser l'information et transforme la CMR en un ensemble d'actions exécutables par l'outil choisi. Inversement, il transforme les informations venant des outils graphiques en expressions CMR qui seront envoyées vers le contrôleur de dialogue.

L'outil de saisie appelé Outil de Dessin (OD) qui permet la saisie des plans de bâtiments et de réseaux sera détaillé dans ce chapitre. Les autres outils appelés outils de représentation graphique tels que les histogrammes, les "camemberts" et les tables sont exposés dans les travaux de Chapel[Chap 91].

La suite de ce chapitre sera consacrée à l'outil de saisie graphique qu'est l'OD (Outil de Dessin). Dans la section 2, nous présenterons les objectifs et les fonctionnalités de l'OD. Dans la section 3, nous décrirons les structures de données et les algorithmes importants utilisés. La section 4 sera consacrée aux échanges entre l' OD et le reste du système. La portabilité de l'OD sera discutée dans la section 5, où nous exposerons la dépendance de ce module vis-à-vis de l'application et du système de fenêtrage. Enfin, dans la section 6 nous conclurons par l'évaluation et les extensions futures du module.

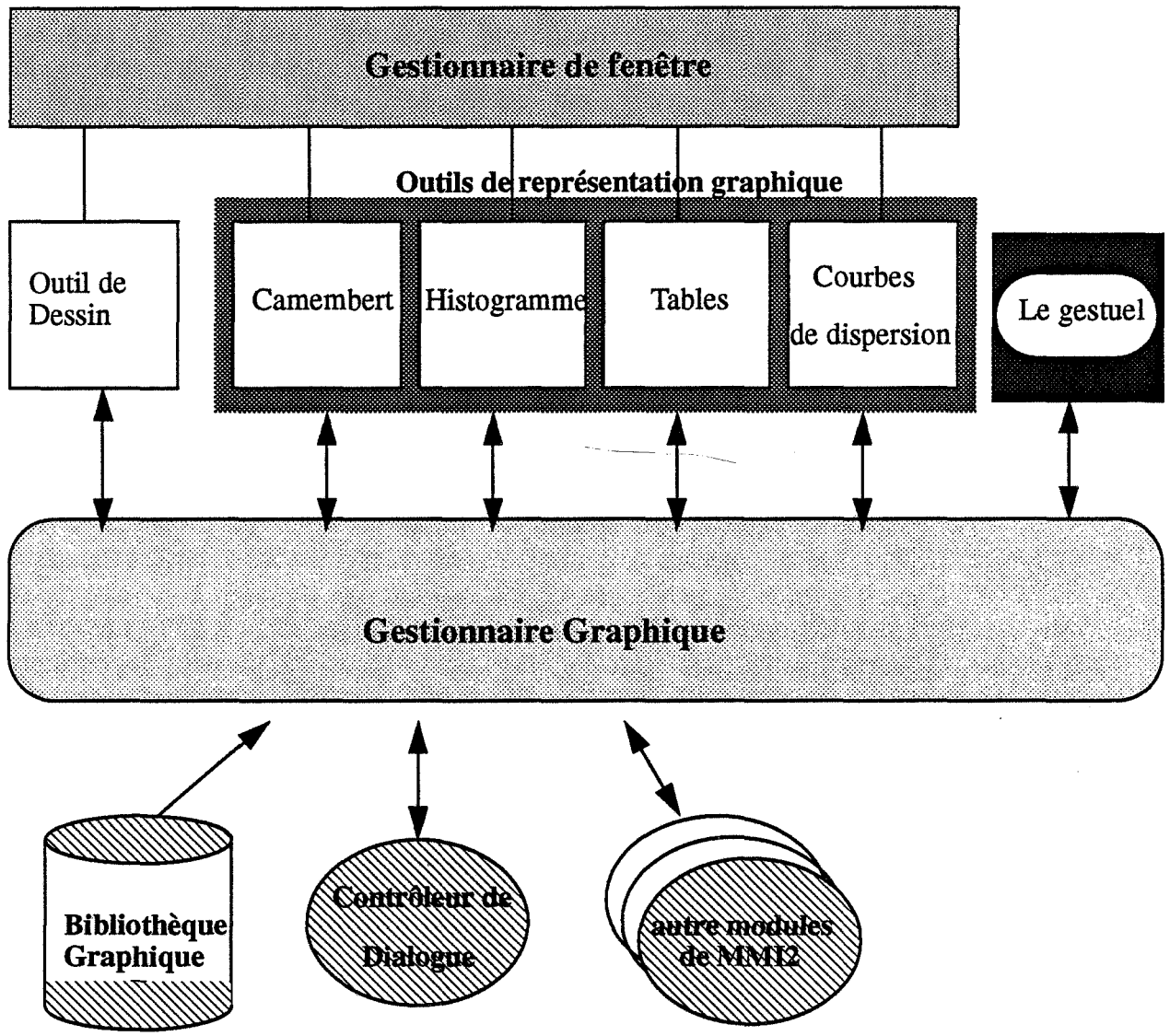


Figure 38: Architecture de l'Expert Graphique

IV-2 L'OUTIL DE DESSIN

L'expérience effectuée au début du projet avec des experts humains [Caho 89] nous a révélé l'importance du graphique dans une activité de conception de réseau. En effet, les experts humains utilisent la langue naturelle en faisant souvent référence à des dessins graphiques simplifiés représentant le réseau et le bâtiment (voir la figure 39). A partir de cette expérience, nous avons établi les objectifs du module graphique qui sont présentés ci-dessous:

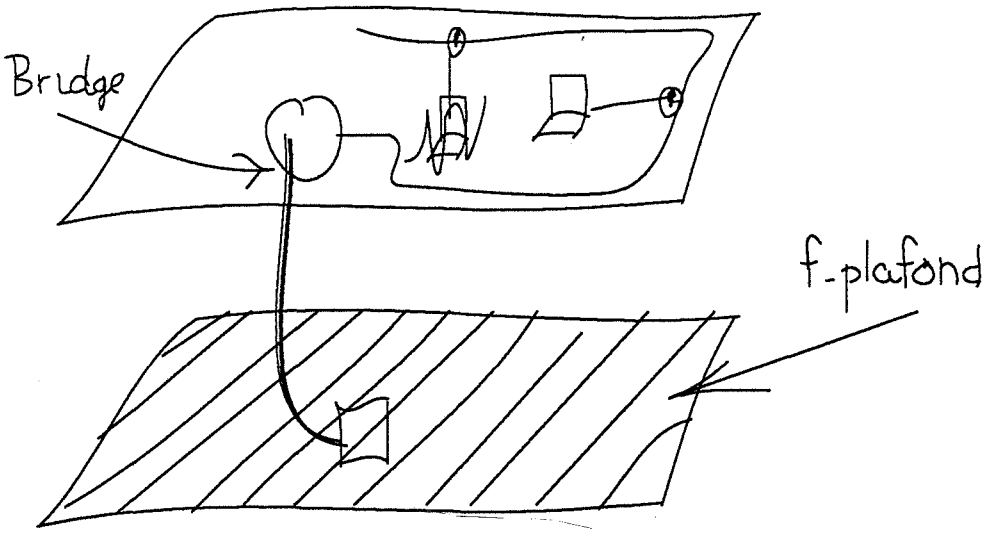


Figure 39: Dessin sur papier d'experts humains

- Permettre à l'utilisateur de saisir graphiquement un ou plusieurs étages d'un bâtiment et un réseau soit à main levée, soit à l'aide d'outils graphiques qui seront décrits ultérieurement. Le bâtiment et le réseau seront représentés par deux cartes planaires distinctes. Cependant, il faut maintenir des liens entre le bâtiment et le réseau: par exemple, telle machine se trouve dans telle pièce, tel câble longe tel mur,... A cet effet, nous avons développé des procédures qui attachent des attributs aux structures de données, permettant ainsi d'établir des liens entre deux cartes planaires.

- Effectuer un prétraitement du tracé (échantillonnage, recalage, élimination des ratures). En effet, le tracé initial présente des malformations qui seront traitées au cours cette étape. Le prétraitement est effectué en trois étapes appelées ECHANTILLONNAGE, RECALAGE et ELIMINATION DES RATURES. L'étape d'ECHANTILLONNAGE peut être résumée en trois points: réduction du nombre de points, correction des déformations et élimination des bavures (voir la section 3). L'étape de RECALAGE prend en entrée la liste des points échantillonnés et effectue un recalage sur des segments de droite. Enfin l'étape d'ELIMINATION DES RATURES est effectuée un peu plus tard, après construction des cartes planaires, car elle nécessite une connaissance préalable du contexte.

- Déduire la sémantique du tracé à partir des structures de données et envoyer ces informations au gestionnaire graphique sous forme de clauses Prolog. Le gestionnaire graphique transformera ces informations en expressions CMR qui seront transférées vers le contrôleur de dialogue.

- Servir de support au module gestuel. Quand l'utilisateur sélectionne dans la fenêtre de l'OD le mode gestuel, tous les événements de la souris seront triés et envoyés vers le module gestuel.

- Permettre une représentation 3D du bâtiment et du réseau. En effet, les cartes planaires sont des structures de donnés représentant un dessin 2D; l'OD les visualise de façon à ce qu'on ait l'impression d'une vue 3D.

- Permettre aux autres modules du système de manipuler des objets graphiques. Pour cela, un ensemble de prédicats a été implémenté permettant ainsi aux autres modules d'accomplir des opérations graphiques sur les objets, par exemple déplacer un objet, détruire un objet,...

- Rendre compte au reste du système MMI2 des actions de l'utilisateur. Quand l'utilisateur accomplit une tâche, l'OD informe le système de l'action de l'utilisateur et, dans certains cas, demande une autorisation pour exécuter cette tâche. Si l'autorisation est refusée l'action ne sera pas exécutée et l'utilisateur sera averti.

IV-2.1 Architecture de l'Outil de Dessin

L'OD communique avec l'utilisateur à travers le système de fenêtrage, et avec l'application à travers le gestionnaire graphique (voir la figure 40). Pour des raisons de portabilité et de flexibilité évidentes, l'OD contient deux modules qui servent d'interfaces, d'une part, avec le gestionnaire graphique et d'autre part, avec le système de fenêtrage. En entrée, les événements de Sunview sont filtrés par le gestionnaire d'événements qui les aiguille soit vers le noyau de l'OD, soit vers le module gestuel. En sortie, le noyau de l'OD se sert d'une bibliothèque graphique qui contient un ensemble de fonctions élémentaires dépendant de Sunview. Ces fonctions permettent de tracer les polygones et de manipuler les fenêtres et les "bitmaps'.

Pour communiquer avec le gestionnaire graphique, nous avons développé un ensemble de prédicats Prolog. Ces prédicats permettent de véhiculer les informations de l'OD vers le gestionnaire graphique et aussi, d'activer les méthodes (procédures) qui agissent sur les structures de l'OD.

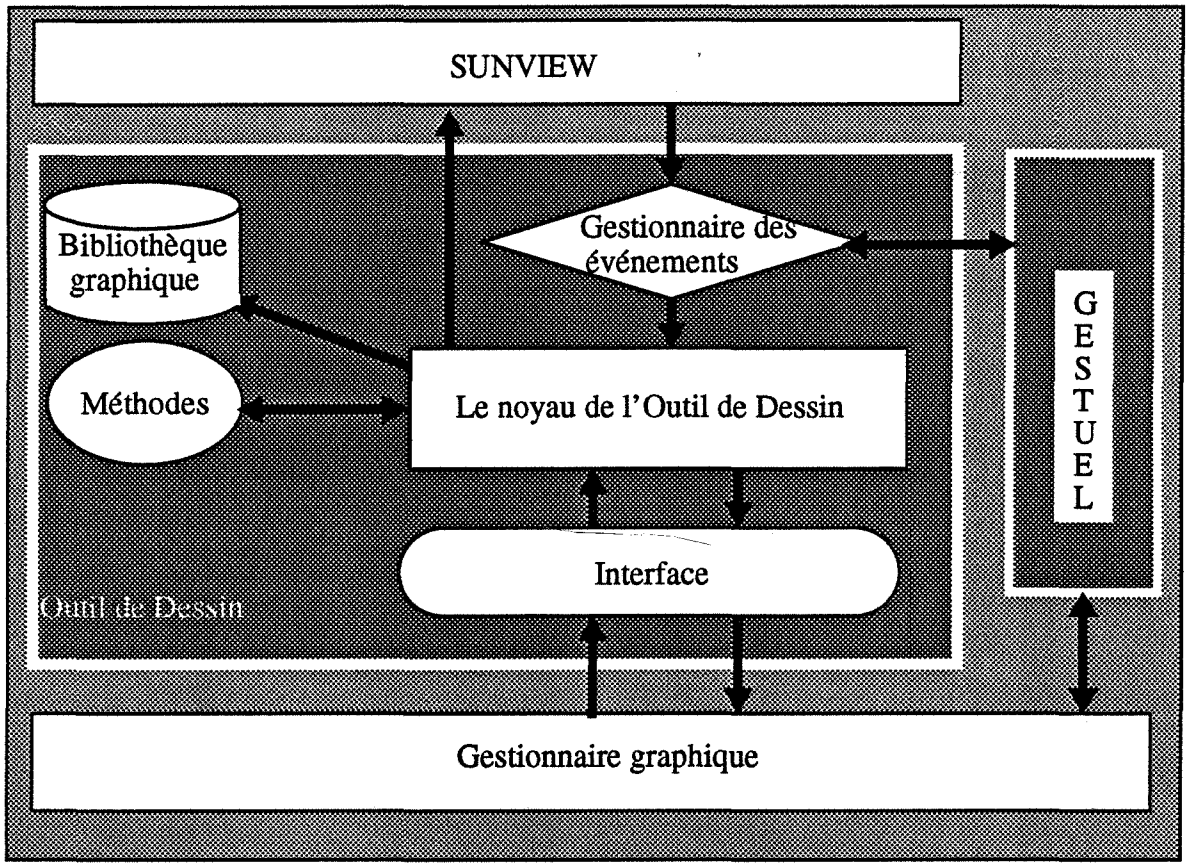


Figure 40: Architecture de l'Outil de Dessin

IV-2.2 Présentation de l'interface de l'OD

L'interface du module graphique (Outil de Dessin) est composée de trois fenêtres (voir la figure 41):

- ♦ Une fenêtre de saisie, qui est affichée initialement et qui contient deux zones:
  - • une zone de dessin, dans laquelle l'utilisateur introduit le plan du bâtiment et installe le réseau. Dans cette fenêtre, on peut afficher soit le bâtiment, soit le réseau, soit les deux à la fois. En haut de cette zone, des informations utiles sont affichés telles que l'identificateur de l'étage courant, la longueur des arêtes, et le type de la carte affichée (voir la figure 42a).

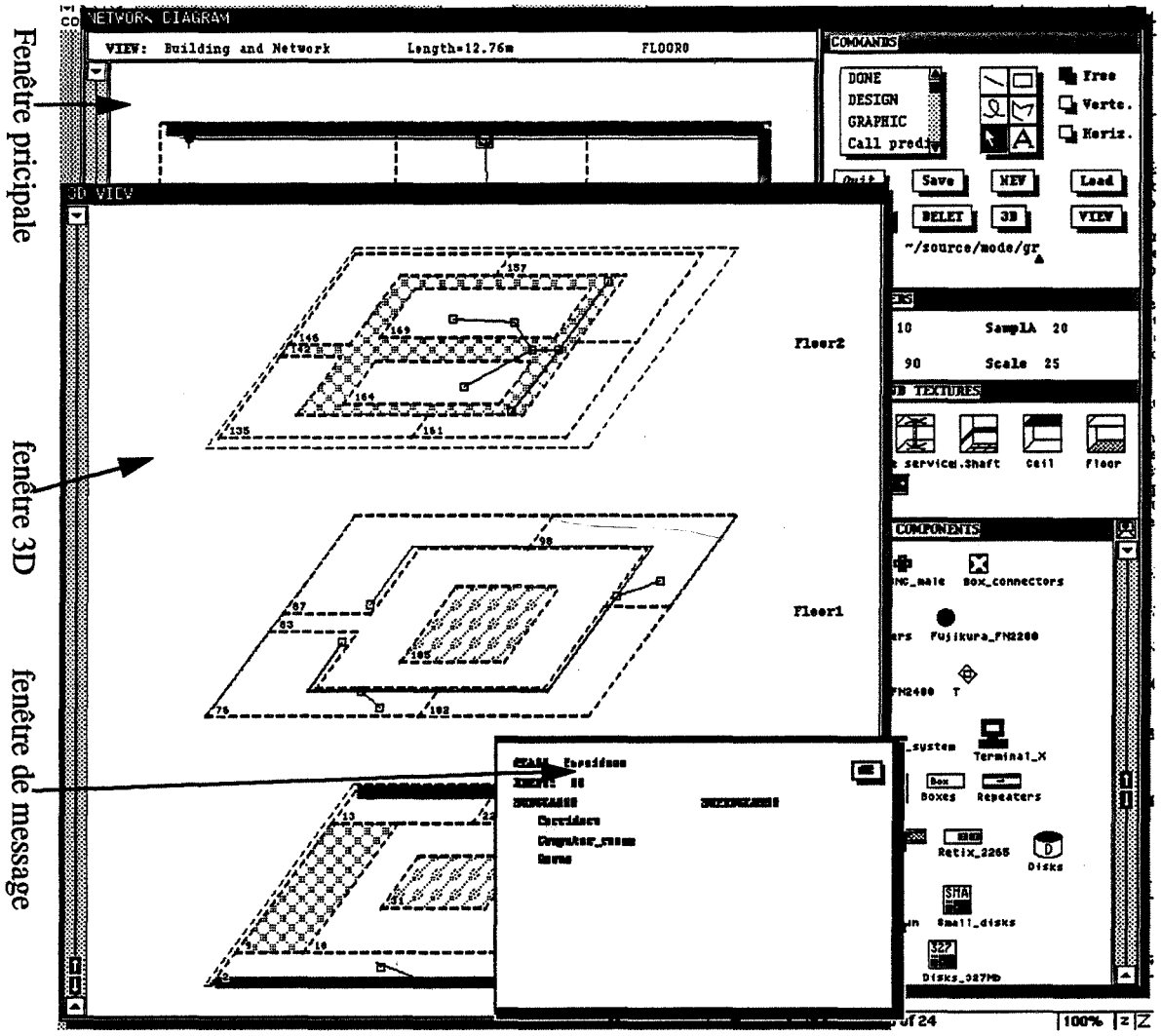
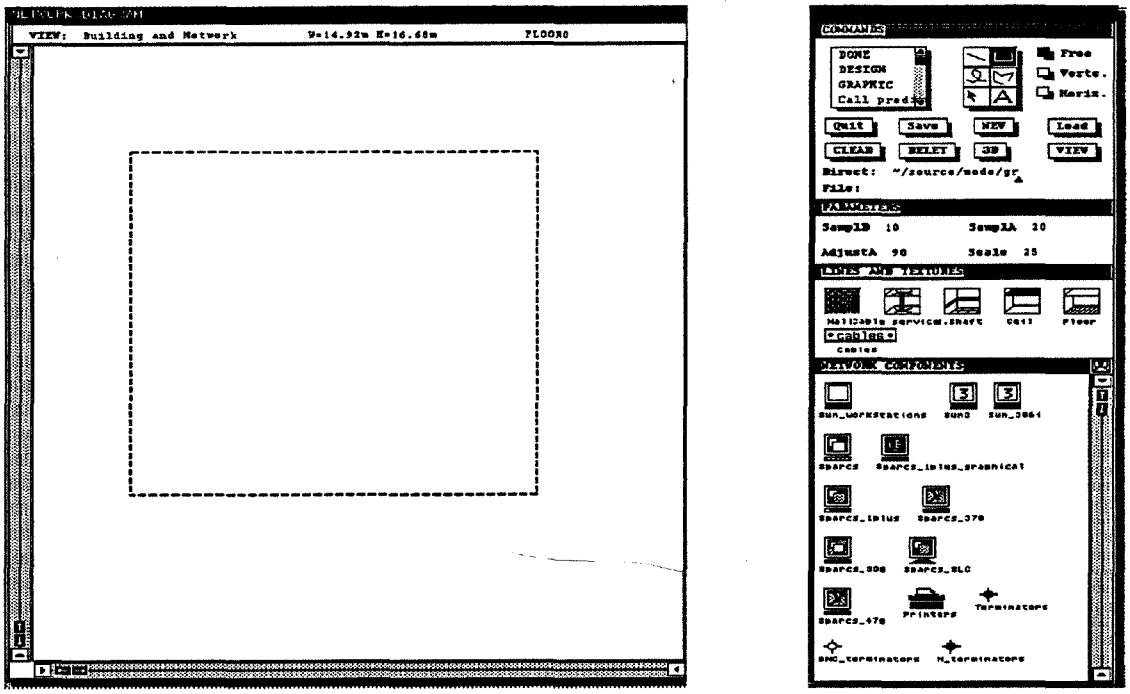


Figure 41: Les fenêtres de l'Outil de Dessin

- une zone de commande (voir la figure 42b), qui contient des boutons, des menus, des paramètres de recalage et des icônes représentant l'équipement matériel du réseau: machines, disques, prises, terminateurs,... Des outils d'aide au dessin permettent à l'utilisateur une saisie plus aisée des plans du bâtiment et du réseau (lignes, boîtes, lignes élastiques, dessin à main levée). De plus, des boutons permettent de contrôler ces outils en imposant un sens vertical ou horizontal aux lignes ainsi tracées.



a:zone de dessin *b:zone de commande*

Figure 42: La fenêtre 2D

- Une fenêtre 3D, qui sera affichée à la demande de l'utilisateur, sur laquelle sont visualisés trois étages consécutifs du bâtiment en simulant une projection d'objets 3D. Un ascenseur permet à l'utilisateur de visualiser tous les étages du bâtiment. Certaines opérations graphiques sont exécutées à partir de cette fenêtre dans laquelle on a une vue globale des étages; par exemple, si l'utilisateur veut installer une gaine verticale, il doit alors indiquer deux points de l'écran situés sur deux étages successifs.

- Une fenêtre de messages permettant au système d'afficher des informations à l'intention de l'utilisateur ou de lui demander un complément d'informations.

IV-2.2.1 Les objets et les opérations

L'OD permet de manipuler des objets qui ont une signification pour l'application dans le sens où ils sont représentés dans l'arbre sémantique de l'Expert Sémantique. Du point de vue de l'utilisateur, ces objets ont une représentation graphique et un certain nombre d'opérations peuvent leur être appliquées.

De point de vue de l'utilisateur, on peut classer les objets en deux classes: les objets du bâtiment et les objets du réseau. En effet, un bâtiment (*Building*) est composé de plusieurs étages et chaque étage (*Floor*) est représenté sur une "page" dans la zone de dessin; de plus, un étage est constitué de plusieurs pièces (*Room*), chacune d'entre elles

étant formée d'une succession de murs (*Walls*). Une pièce peut être de trois types: une pièce "ordinaire", un corridor ou une salle machines (*Computer room*). Un mur peut être de deux types: perforable ou non perforable, qui sont représentés par deux textures différentes. De gaines techniques (*Horizontal_shaft*) peuvent être installées le long des murs. De faux plafonds (*False_ceiling*) et de faux planchers (*False_floor*) peuvent être installés dans les pièces; chacun de ces objets est représenté par une texture. Enfin, en se servant de la représentation 3D, l'utilisateur peut installer des gaines verticales (*Lifts, Stairs, Vertical_shaft*) qui servent au passage d'un câble entre deux étages. Ces gaines sont de trois types: escalier, ascenseur ou "puits".

Les objets du réseau sont les câbles et les équipements matériels du réseau. Cependant la classe des câbles (*Cables*) a deux sous-classes qui sont la sous-classe des câbles fins (*Thin_cables*) et celle des câbles épais (*Thick_cables*). Chaque classe de câble est représentée par une ligne de texture spécifique. Les équipements matériels du réseau sont des objets qui ont une représentation icônique, chaque icône représentant une classe d'équipement; par exemple, sélectionner l'icône disque revient à créer une instance de la classe *(Disks)*.

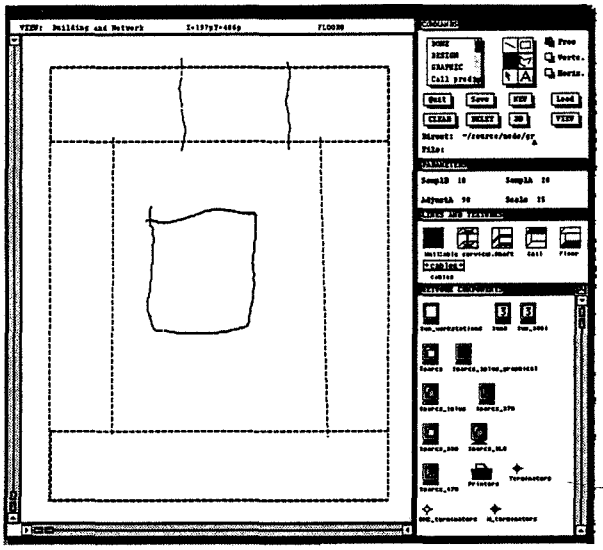
Dans MMI2, chaque objet a un identificateur unique permettant aux différents modules du système de le désigner de façon non ambiguë. Ainsi, lors de la création d'un objet, l'OD demande au système un identificateur unique pour désigner l'objet qui vient d'être créé.

Voici une description rapide de l'utilisation de l'outil OD. Dans un premier temps, l'utilisateur saisit le plan du bâtiment et du réseau à l'aide de la souris (il pourra éliminer le dernier tracé on utilisant la touche du clavier "Delete"). Estimant le tracé sous sa forme brute satisfaisant (voir la figure 43), l'utilisateur déclenche alors la construction des cartes planaires en sélectionnant, dans le menu de la zone de dessin, l'item "Construct PM". Le plan du bâtiment (resp. du réseau) est alors restitué sous une forme agréable, où les segments de droites ont été échantillonnés, recalés et les "ratures" éliminées. Les paramètres d'échantillonnage (SamplD, SamplA) et de recalage (AdjusA) peuvent être modifiés, au préalable, par l'utilisateur. Les segments de droite ainsi obtenus sont interprétés comme représentant les murs du bâtiment (resp. les cables du réseau); de même, les faces du bâtiment sont supposées être des pièces. Le système associe alors, de manière incrémentale, des identificateurs aux objets du bâtiment et du réseau. Pour des raisons de commodité, les identificateurs de pièce (Room1,Room2...) sont affichés en bas et à gauche de chaque pièce.

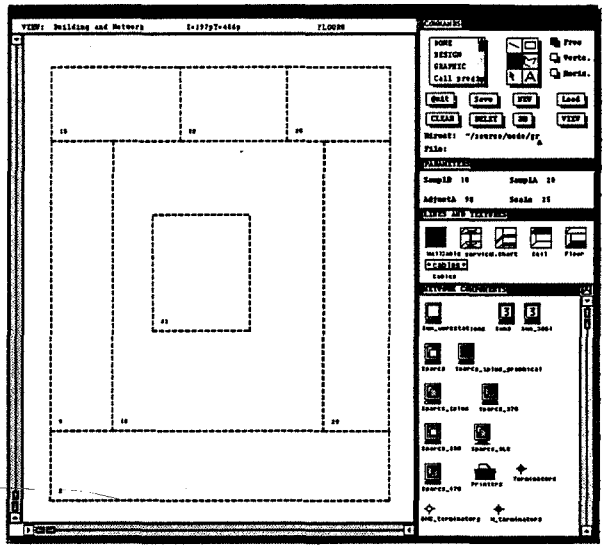
Tout ajout ou destruction d'arêtes survenu après l'activation de l'item "Contruct PM" entraîne la mise à jour de la carte planaire de manière incrémentale.

Une fois les câbles installés (ces câbles doivent être nécessairement à l'intérieur des pièces), l'utilisateur met en place les composants matériels du réseau (stations de travail, disques,...) dans les pièces du bâtiment. Il faut remarquer que les connecteurs (prises, terminateurs, transceiveurs) doivent être assez proches des câbles. En effet, chaque connecteur est recalé sur le câble le plus proche de façon à ce que l'ensemble des

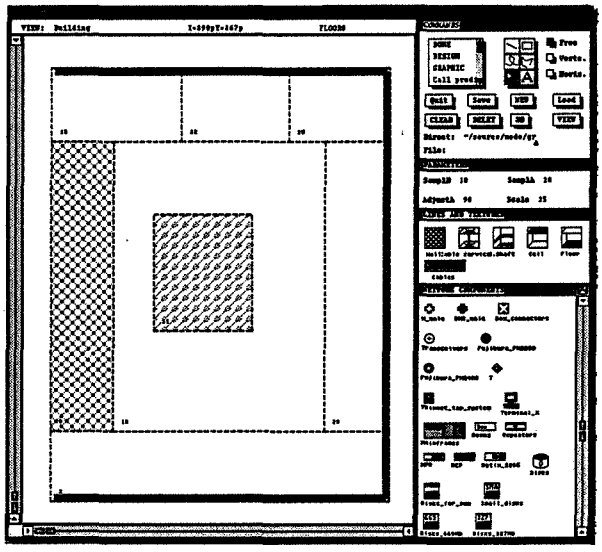
segments séparant deux connecteurs soit considéré comme étant un câble du réseau.



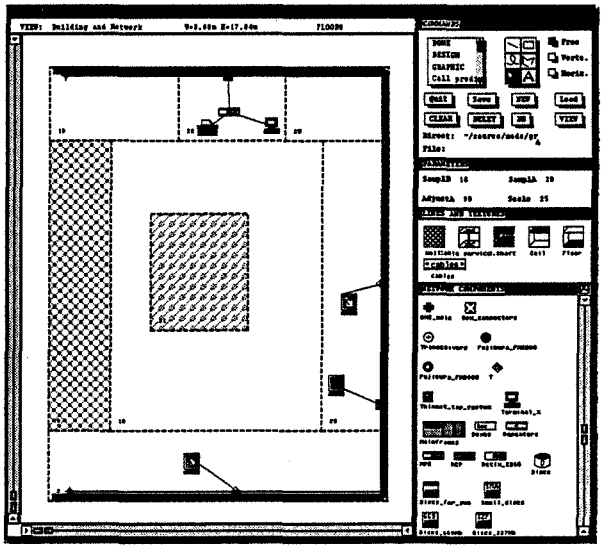
tracé initial



Prétraitement et Construction de la carte planaire



Installation du faux-plafond, du faux-plancher et de la gaine technique



Installation du réseau

Figure 43: saisie des plans du bâtiment et du réseau

Le module OD offre à l'utilisateur plusieurs commandes permettant de contrôler l'interface et l'affichage des objets graphiques. Ces commandes se présentent sous forme de boutons ou de menus contenus dans la zone de commandes et qui sont décrits dans l'annexe A.

Nous verrons plus loin que tous ces objets ont une structure de données arborescente et que l'utilisateur ou le système peut modifier la classe d'une instance d'objet. L'utilisateur peut manipuler ces objets en leur appliquant les opérations suivantes:

- *La création d'un objet*: l'action que doit accomplir l'utilisateur pour créer un objet est différente selon le type de l'objet. En effet, si l'utilisateur sélectionne le bouton "Wall" et saisit un tracé dans la zone de dessin, l'OD crée les nouveaux objets (Walls, Rooms). Les objets "Walls" peuvent être de type "Perforable" ou "non Perforable" selon le choix fait préalablement par l'utilisateur grâce au menu associé au bouton "Wall". Le même processus permet la création des câbles et des gaines techniques (voir la figure 44). Ces objets sont représentés par des segments de droite ayant une épaisseur et une texture spécifique.

  Les objets tels que les faux-plafonds (*False_ceiling*) et le faux-planchers (*False_floor*) sont créés en sélectionnant préalablement une pièce, puis en désignant le bouton associé à l'un de ces objets. La représentation graphique de ces objets est une texture spécifique qui recouvre toute la pièce sélectionnée (voir la figure 44).

  La création d'un objet de type équipement matériel du réseau se fait par sélection d'une icône dans la zone de commande et par localisation de cette icône dans la zone de dessin.

  Enfin, pour créer un nouvel étage (*Floor*) il suffit de désigner le bouton "NEW". Une nouvelle "page" est affichée dans la zone de dessin permettant la saisie d'un nouvel étage. Actuellement, un seul objet (*Building*) et un seul objet (*Network*) sont créés automatiquement par l'OD au début de la session.

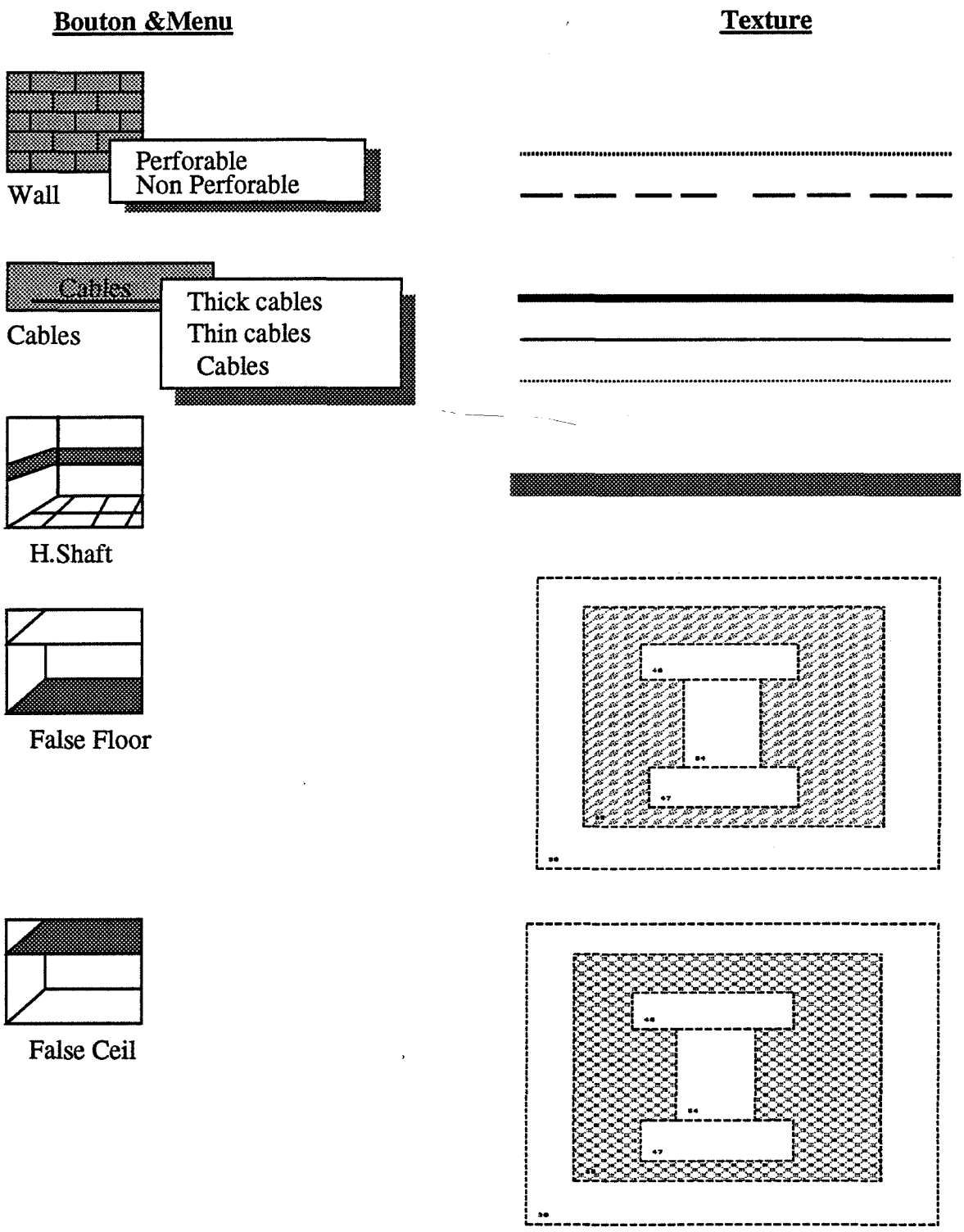


Figure 44: création d'objets

♦ *La destruction d'un objet:* il est essentiel que l'utilisateur puisse détruire les objets qu'il a créés ou que le système a créés. Il suffit que l'utilisateur sélectionne l'objet en question, puis choisisse dans le menu de la zone de dessin l'item "Delete". On remarquera que si l'objet est lié à d'autres objets, tous ses liens seront également détruits.

♦ *La sélection d'un objet*: tous les objets affichés dans la zone de dessin peuvent être sélectionnés par l'utilisateur sauf les objets bâtiment (*Buildings*), étage (*Floors*) et réseau (*Networks*). En général, pour sélectionner un objet, il suffit de se mettre en mode sélection en désignant la "flèche" (voir la figure 45), en se plaçant sur l'objet et en appuyant sur le bouton gauche de la souris: l'objet sélectionné sera affiché en inverse vidéo.

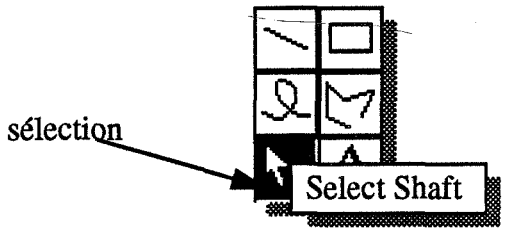


Figure 45: Sélection

♦ *Le déplacement d'un objet:* seuls les objets qui ont une représentation icônique peuvent être déplacés. Pour déplacer une machine, l'utilisateur doit désigner l'objet avec le bouton du milieu de la souris, puis, tout en maintenant le bouton de la souris en position "bas" déplacer l'objet jusqu'à sa nouvelle place et enfin "lâcher" le bouton de la souris. Si l'objet déplacé est lié à d'autre objets, ses liens seront mis à jour.

♦ *La connexion d'objets:* dans la conception des réseaux informatiques, on connecte souvent des objets du réseau entre eux. Par exemple, une station de travail peut être connectée à un disque, ou un terminateur peut être connecté à une extrémité de câble. Certaines de ces connexions sont simples (par exemple connecter un disque à une station de travail); d'autres sont plus complexes, la connexion entre deux objets faisant intervenir d'autres objets. Par exemple, la connexion entre une prise (*Thinnet_tap_system*) et une station de travail (*Workstation*) crée automatiquement deux objets: un transceiveur (*Transceiver*) et un câble AUI (*AUI_twin_coaxial_cable*) et connecte la station avec le transceiveur, le transceiveur avec le câble AUI et enfin le câble AUI avec la prise.

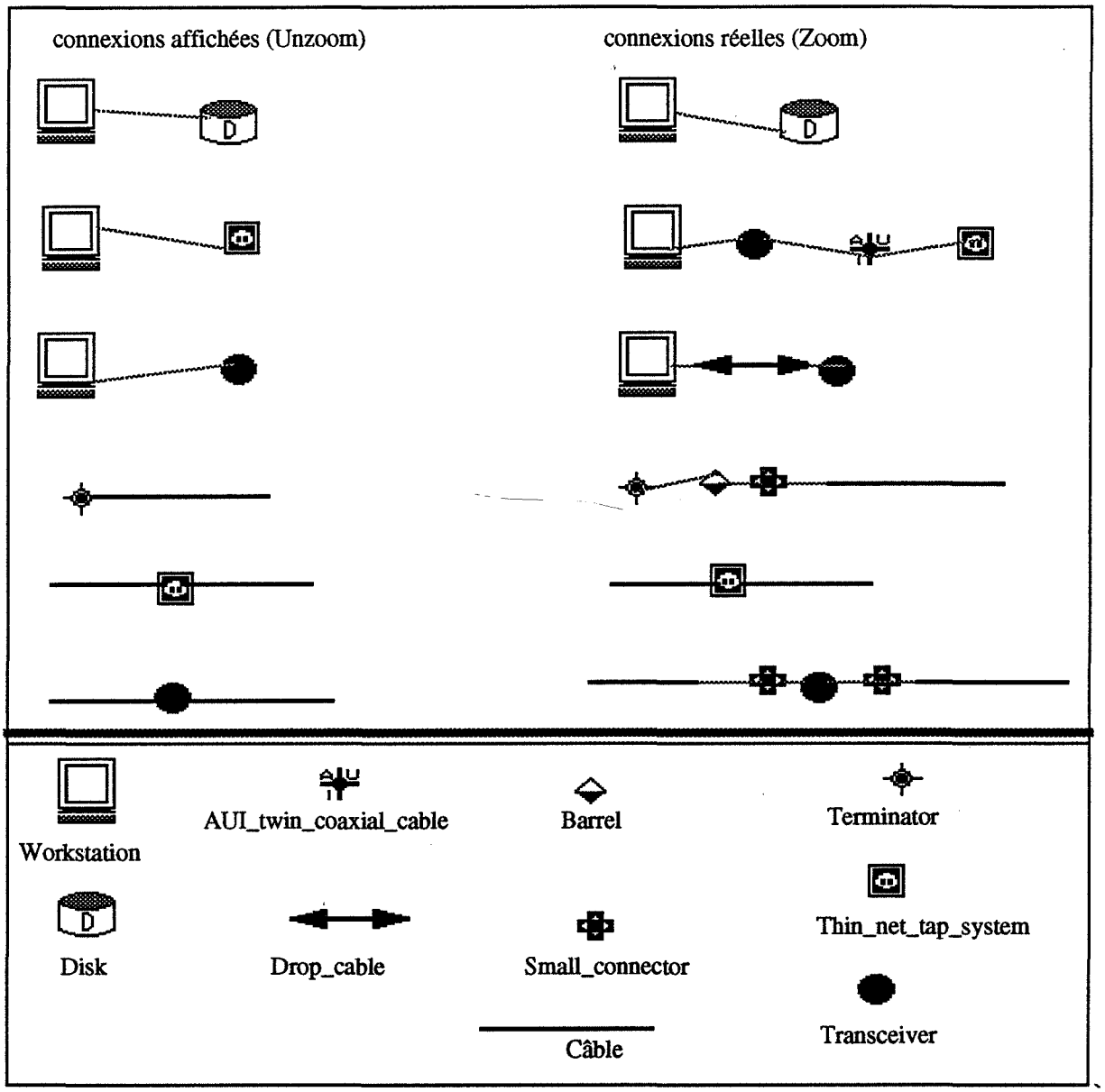


Figure 46: Les connexions

Une connexion entre deux objets peut se faire de deux façons: la première est réalisée automatiquement par le module OD. Généralement, les connexions entre les connecteurs (les terminateurs, les prises et les transceiveurs) et le câble se font automatiquement lorsqu'on insère un nouveau connecteur. Lorsqu'on insère une prise ou un transceiveur, le module cherchera le câble le plus proche, l'éclatera en deux câbles et enfin connectera les deux câbles au transceiveur ou à la prise. Par contre, lorsqu'il s'agit d'un terminateur, le module cherchera l'extrémité la plus proche du câble et connectera le terminateur à cette extrémité. Dans les deux cas, la connexion avec le câble se traduit par la projection de l'objet sur le câble. Le deuxième type de connexion est fait à la demande de l'utilisateur qui doit

sélectionner préalablement les deux objets et activer la commande "link" du menu de la zone de dessin. S'il existe plusieurs façons de connecter les deux objets, la fenêtre de message est affichée pour demander à l'utilisateur de préciser le type de la connexion (voir la figure 47).

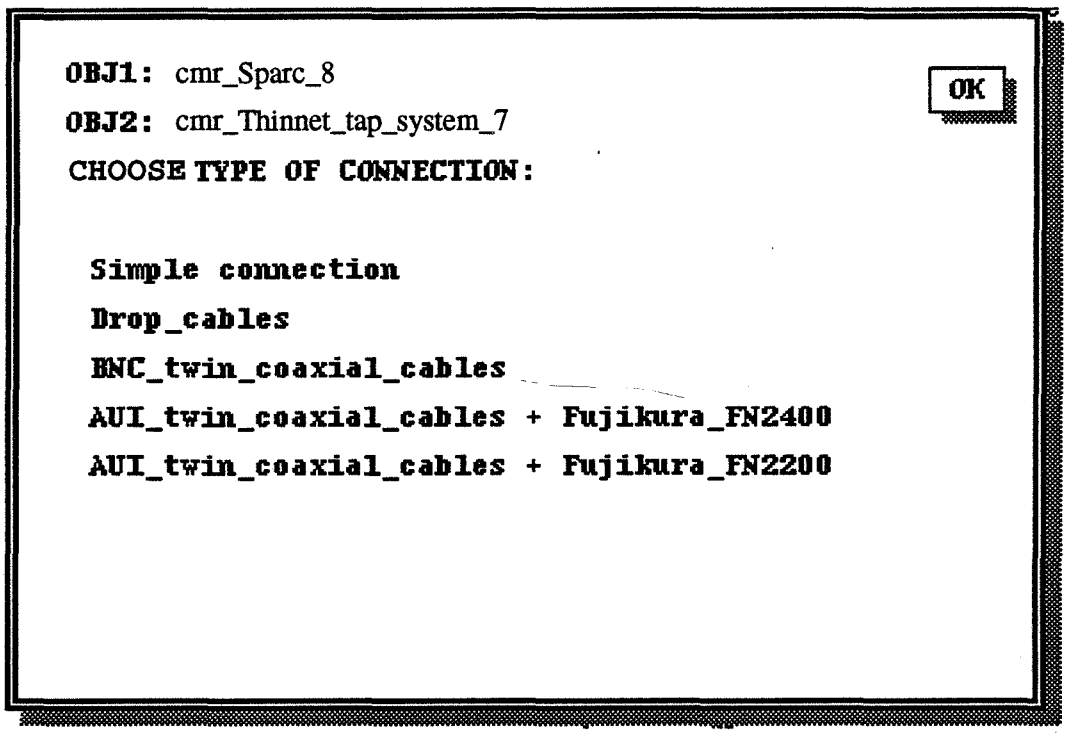


Figure 47: L'utilisateur doit préciser le type de la connexion

Toutes les connexions se traduisent graphiquement sur l'écran par un segment de droite liant les objets initiaux, sauf les connexions avec le câble qui se traduisent par la projection de l'objet sur le câble le plus proche.

Les objets créés automatiquement lorsqu'on connecte deux objets ne sont pas visibles sur l'écran. Pour visualiser ces objets, l'utilisateur doit sélectionner l'item "Zoom" du menu de la zone de commande.

Enfin, l'utilisateur peut sélectionner plusieurs objets à connecter et activer la procédure de connexion. Le système connecte les objets ainsi sélectionnés deux par deux.

- *La déconnexion d'objets:* pour détruire une connexion entre deux objets, il faut sélectionner les deux objets et activer la commande "Unlink" du menu de la zone de dessin.

- *La reclassification d'objet:* Souvent, l'utilisateur manipule d'abord un objet assez "général", par exemple un ORDINATEUR, avant d' affiner la définition de l'objet en disant qu'il s'agit d'un SERVEUR ou d'un SUN4. Il faut donc donner à

l'utilisateur les moyens de désigner, à l'aide d'icônes, les objets (ORDINATEUR, SERVEUR,.....), et la possibilité d'affiner la définition de l'objet. En sélectionnant l'item "Reclassify" dans le menu de la zone de dessin après avoir sélectionné l'objet, la fenêtre de message affichera la classe de l'objet et les classes auxquelles il peut appartenir, de la plus précise à la plus générale (voir la figure 48).

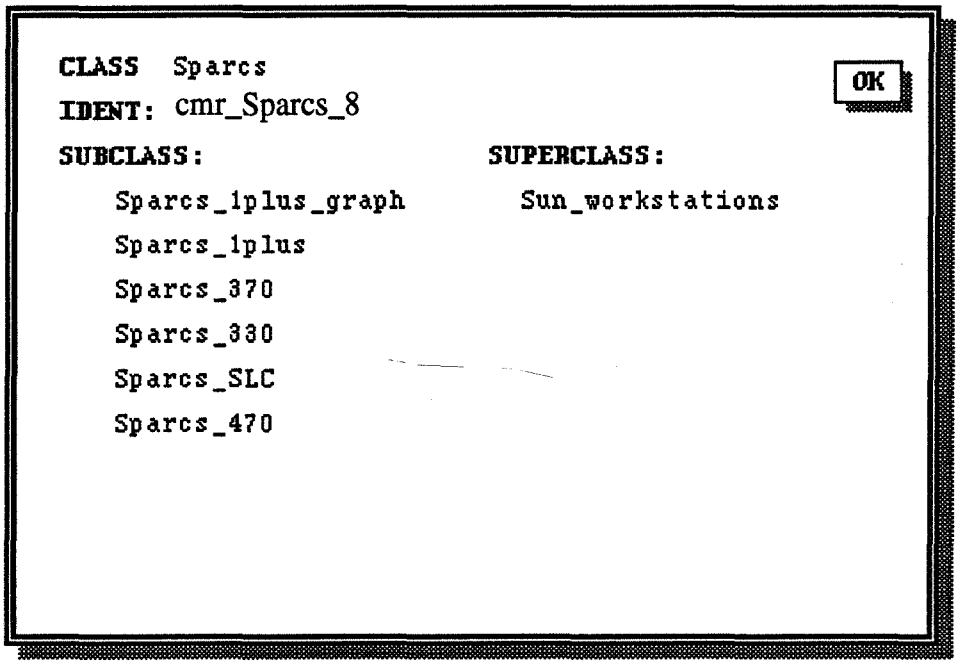


Figure 48: Reclassification d'un objet

Aussitôt que l'utilisateur a redéfini l'objet en sélectionnant sa nouvelle classe, la représentation graphique de l'objet change.

- *Le nombre de connexions:* On a remarqué que lors de la conception d'un réseau, les experts associent dans un premier temps des informations complémentaires aux pièces. Parmi ces informations, on trouve le nombre de connexions dans une pièce: l'utilisateur doit indiquer au système le nombre de connexions qu'il désire avoir dans une pièce. Pour cela, il doit sélectionner la pièce et activer l'item "Wiched connections". Dans la fenêtre de message, il doit introduire le nombre de connexions et valider par le bouton "OK" (voir la figure 49).

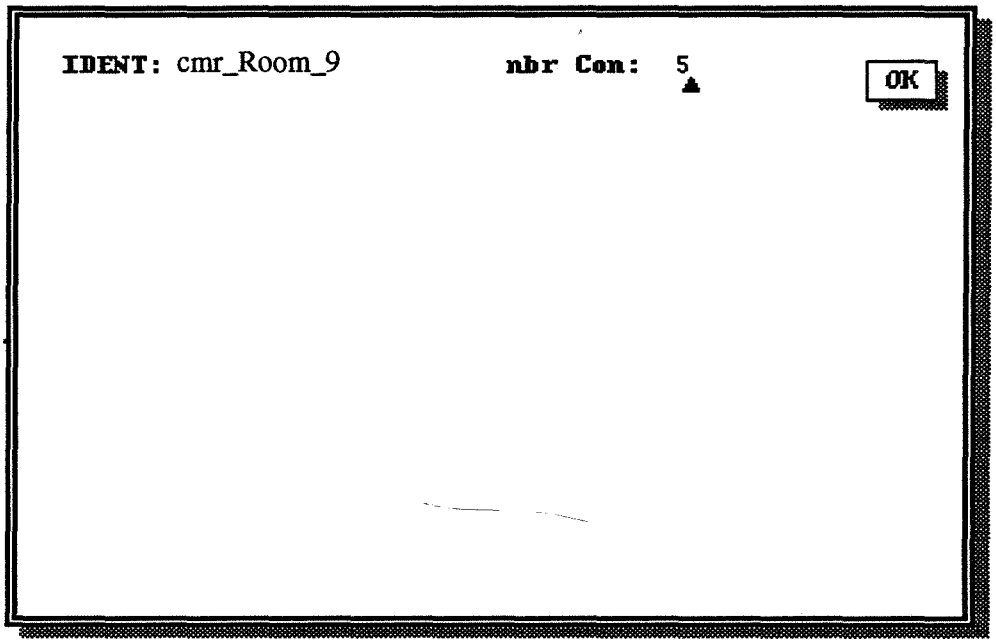


Figure 49: "Wiched connections"

- *La départementalisation:* Une autre information complémentaire utilisée par les experts est la départementalisation (un département) est un ensemble de pièces ayant la même fonction). L'utilisateur doit alors sélectionner la ou les pièces qui seront considérées comme un département et activer l'item "Department".

- *L'affichage des informations:* L'utilisateur manipule un certains nombre d'objets graphiques à travers plusieurs modes et il doit parfois les désigner par leur identificateur. Cependant, l'identificateur et d'autres informations sur un objet ne sont pas visibles sur l'écran. On a prévu une commande qui permet à l'utilisateur d'afficher les informations concernant les objets, telles que son identificateur et les objets auxquels il est connecté (voir la figure 50). Il suffit à l'utilisateur de sélectionner l'objet et d'activer l'item "Get info"; il verra apparaître la fenêtre de message contenant les informations concernant l'objet.

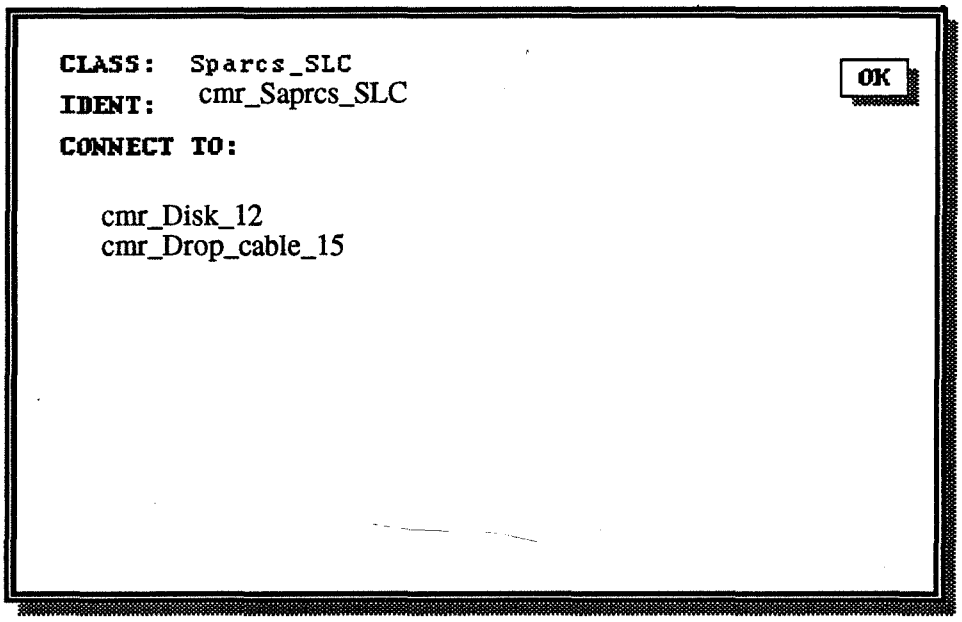


Figure 50: "Get Info"

IV-3 STRUCTURES DE DONNEES ET ALGORITHMES

Pendant la réalisation du module OD trois problèmes se sont posés:

1- Quelles structures de données fallait-il choisir pour représenter la topologie et la géométrie du tracé?

Une première solution est une structure de données dynamique décrivant les configurations du plan [Over 81]. Cette structure permet d'entretenir efficacement l'enveloppe convexe d'un ensemble de points du plan, subissant des insertions et des suppressions dynamiques. Cette structure n'offre apparemment pas de support pour la topologie du tracé.

Une structure de données plus récente [Ayal 85], est celle d'arbre quaternaire exact. Cette structure est un arbre représentant un polygone; la racine est le carré contenant le polygone; si le contenu du carré est trop complexe, il est coupé en quatre parties égales, filles du carré initial, jusqu'à obtention de noeuds terminaux. Il existe cinq types de noeuds terminaux:

- les feuilles-sommets, qui contiennent un seul sommet et les arêtes incidentes
- les feuilles-arêtes, qui sont traversées par une arête unique
- les feuilles blanches, qui sont totalement en dehors du polygone
- les feuilles-noires, qui sont totalement à l'intérieur du polygone
- les feuilles pixels, qu'il est inutile de diviser encore.

Cette structure de données semblait être intéressante, mais ses performances sont moins bonnes que celles des cartes planaires.

La carte planaire est une représentation plane d'un graphe planaire. Elle est constituée de deux parties: une carte planaire locale est une partition du plan topologique en sommets , en arête et en faces; et d'une carte planaire globale qui traduit plutôt la géométrie du tracé (voir le chapitre III).

2- La saisie à main levée est sujette à des ratures et des malformations.

Il est donc indispensable de prévoir un processus permettant de supprimer le maximum de ratures et de bavures avant la construction de la carte planaire. Sans compter que, pour l'utilisateur, il est agréable de se voir restituer un dessin bien construit sans avoir à soigner excessivement le tracé.

3- La déduction de la sémantique.

L'OD doit transmettre toutes les informations (topologiques et géométriques) au reste du système. Il a été convenu que la CMR soit le support de communication entre les différents experts du système. Comme le gestionnaire graphique joue le rôle d'intermédiaire entre les outils graphiques et le reste du système, l'OD transmettra les informations sous forme de clauses Prolog qui seront traduites par le gestionnaire graphique en expressions CMR.

En réalité, la saisie des données et la construction de la carte planaire se font en trois étapes: *saisie du tracé, prétraitement du tracé* et *construction de la structure de données.*

IV-3.1 La saisie

Nous utilisons la souris comme moyen d'acquisition; celle-ci envoie les coordonnées des points ainsi tracés. Le lever du bouton de la souris et "l'appui" sur ce dernier, provoquent l'émission d'un événement spécial. Le résultat d'une saisie se présente donc sous la forme d'une liste de points étiquetés par la levée ou l'abaissement du bouton de la souris. Un élément de la liste est défini comme suit:

```
Structure POINT {
entier x,y;
booleen Lever;
structure POINT suivant;
}
```

Ce mode d'acquisition présente l'avantage de restituer fidèlement le sens du tracé et l'ordre dans lequel ont été dessinées les différentes parties de la figure. Un périphérique comme la tablette graphique sera facile à intégrer et nécessitera peu de modifications de ce qui a été réalisé.

IV-3.2 Le prétraitement

Les malformations d'un tracé à main levée peuvent être groupées en trois catégories. La première comprend les erreurs de tracé et les grosses ratures qui se produisent lorsque l'utilisateur "rate son coup". Elles sont éliminées manuellement pendant le tracé. La seconde catégorie regroupe les erreurs dues au tremblement de la main de l'utilisateur, lors de l'acquisition de la forme brute du dessin et qui seront appelées *bavures*. Leur traitement se fait en deux étapes: on commence par la réduction des points saisis (ECHANTILLONNAGE), puis on recale les segments ainsi obtenus selon un angle donné tout en conservant son milieu (RECALAGE). La troisième catégorie est constituée par la rupture d'un tracé ou la superposition de plusieurs tracés; ces malformations, qui seront appelées des "ratures", ne peuvent être corrigées sans connaissances contextuelles comme l'a souligné [Herot 76]. Le même problème se pose avec les interruptions de tracé; cette dernière catégorie sera prise en compte lors de la construction de la carte planaire.

Il faut remarquer que dans ce travail, nous ne traitons que les segments de droites. Le traitement des courbes est beaucoup plus compliqué, et certaines méthodes présentées ci-dessous déforment complètement l'allure des courbes.

IV.3.2.1 L'échantillonnage

La tablette graphique ou la souris transmet un nombre important de points pour chaque tracé, alors qu'un nombre restreint suffit pour en exprimer l'allure. Il existe plusieurs techniques permettant l'échantillonnage d'un tracé brut qui ont été développées par Berthod et Jancenne [Bert79]. Ils utilisent une méthode de lissage qui, dans certains cas, peut éliminer des points anguleux. Nous avons choisi la méthode exposée dans [Bela 82]. Ainsi, la suite de points représentant un segment de droite peut être réduite à ses extrémités A et B (voir la figure 51)

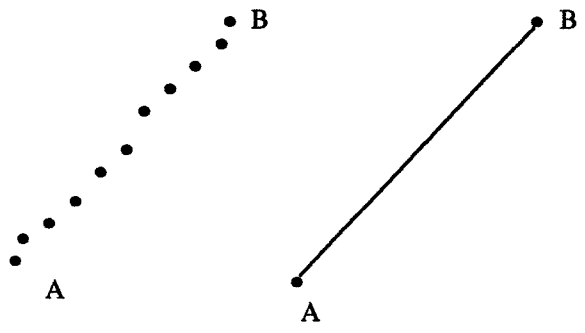


Figure 51: Un segment de droite avant et aprés échantillonnage

Cependant, pour une ligne de direction donnée, les points qui la composent ne sont que rarement alignés. Des déformations (pics) sont introduites involontairement

par l'utilisateur. La correction de ces pics doit se faire en conservant les changements de direction volontaires de l'utilisateur (voir la figure 52).

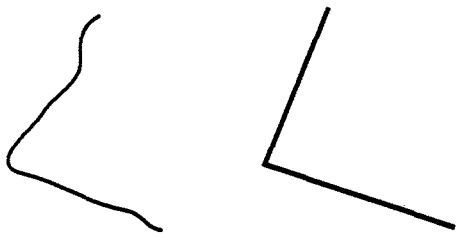


Figure 52: Tracé déformé avant et après échantillonnage

Considérons un tracé A0,A1,A2,... AN. L'algorithme proposé par Belaid [Bela 82] se présente comme suit:

```
X0 ← A0;
X1 ← A1;
X2 ← A2;
Pour i 2 à n faire
{
L  ←angle(X0 X1)
ANGLE  ← angle(V, X1 X2)
si ANGLE >SANGLE et L > SLONG
 alors {/* le point intermédiaire est conservé*/
        X0← X1;
        X1 ← X2;
         V  ←X0 X1;
        X2 ← Ai+1;
        }
sinon {
                /* le point intermédiaire est supprimé*/
        X1 ← X2
        X2  ←Ai+1;
        supprimer Ai-1 de la liste d'entrée;
        }
}
```

SANGLE est l'angle minimum toléré entre deux traits consécutifs. D'après l'étude de Belaid et Masini, une valeur comprise entre 15 et 20 degrés donne de bons résultats.

SLONG est la longueur minimum tolérée pour un trait.

Le tracé ainsi obtenu, est donc transmis à la phase suivante sous forme de

vecteurs joignant les points conservés entre chaque lever de bouton de la souris.

IV-3.2.2 Le recalage

Dans un dessin à main levée, il est important d'assurer le parallélisme ou l'orthogonalité des segments de droite. La procédure de recalage remplace un segment de droite par un autre segment ayant la direction fixée la plus proche de la direction du segment initial, le milieu du segment étant conservé (voir la figure 45).

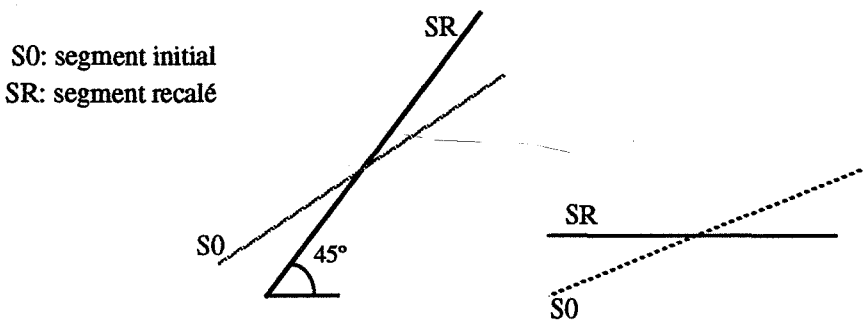


Figure 53: Recalage à 45 degré

Voici la procédure de recalage telle qu'elle a été décrite par Michelucci [Mich84].

recalage
{
/* On note Mi les points d'un tracé, 1 ≤ i ≤ m */
/* On note Ni les points déjà recalés, 1 ≤ j ≤ n */

calculer l'équation de la droite D_2, passant par le milieu de M_1M_2, dont la direction est la direction fixée la plus proche de celle de M_1M_2.
N_1 ← projection_orthogonale(M_1,D2);
j ←1;
Davant ← D_2;
Pour i de 2 à n-1
calculer l'équation de Daprés, droite passant par le milieu de M_i M_{i+1}, dont la direction est la direction fixée la plus proche de celle de M_i M_{i+1};
si Daprés n'est pas parallèle à Davant
alors {
j ← j+1;
N_j ← Daprés∩ Davant;

Davant ←Daprés
}
m←j+1;
traiter N_n comme N_1;
}

IV-3.2.3 L'élimination des ratures

Les types de ratures observés le plus fréquemment sont représentés sur la figure 54; en effet, il est pratiquement impossible de dessiner à main levée une figure avec des connexions parfaites. Corriger de telles ratures, sans connaissances contextuelles, peut être hasardeux. C'est pourquoi, dans le cadre du projet MMI2, ces corrections seront effectuées lors de la construction de la carte planaire.

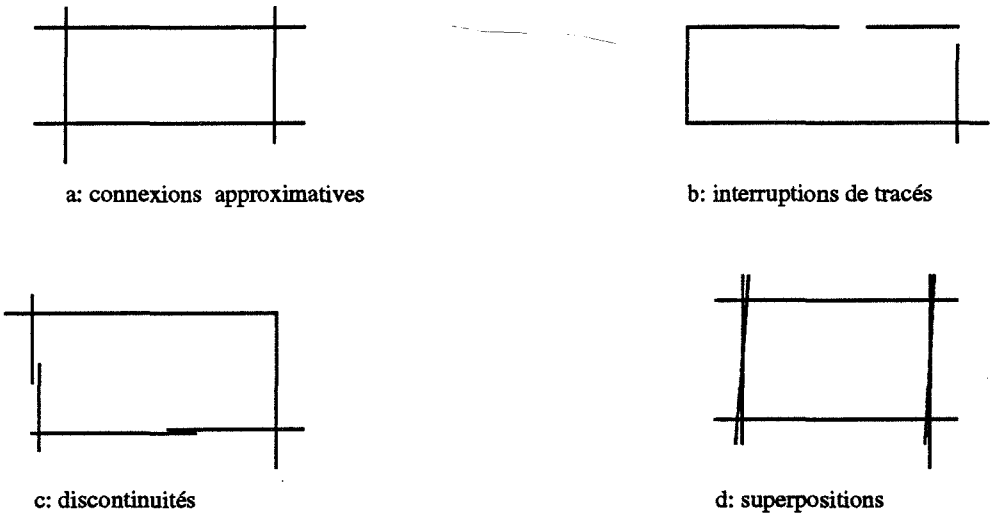


Figure 54 : Exemples de ratures

Malheureusement, il n'existe pas de règles permettant la correction de ce type de ratures. A défaut de règles, nous avons utilisé des méthodes heuristiques.

La construction de la carte planaire nous permet, pour le premier type de rature (voir la figure 55a), d'éliminer, systématiquement ou par sélection, les bouts de segments appelés arêtes pendantes. Pour les types (c) et (d), la carte planaire permet d'éliminer les arêtes superposées et de fusionner les arêtes de discontinuitées.

Le type (b) peut être validé en considérant un cercle de rayon fixé, où toutes les extrémités des segments se trouvant à l'intérieur de ce cercle seront raccordées en un point situé dans le cercle (voir la figure 55). Cette méthode peut être appliquée aussi pour la correction des ratures de type (a).

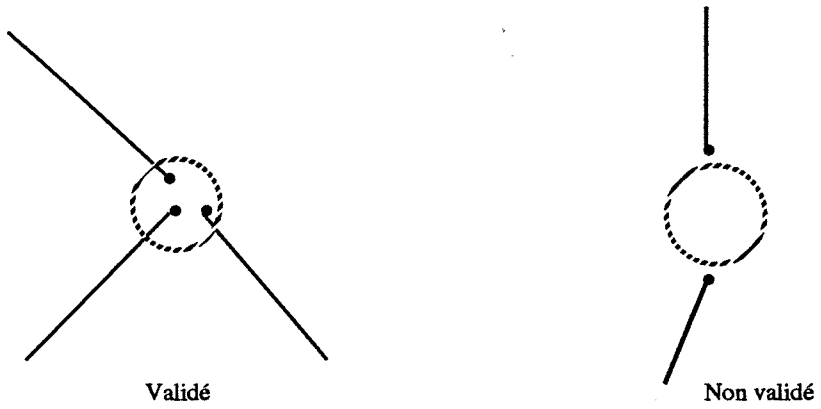


Figure 55: Validation des corrections

Remarquons que le recalage en direction a des effets pervers dans le cas où on veut tracer des courbes. Une solution à ce problème consiste à activer l'algorithme de recalage uniquement pour les segments de droite, et à activer un algorithme basé sur les B-splines pour les courbes.

IV-3.3 La construction des structures de données

La structure de données principale est une liste doublement chainée où chaque noeud représente un bâtiment. Le noeud d'un bâtiment contient une liste doublement chainée où chaque noeud représente un étage. Le noeud de l'étage contient la liste de tous les objets qui constituent l'étage (les pièces, les murs, les gaines techniques, les câbles et les équipement matériels) (voir la figure 56).

Un des objectifs principaux du l'OD est la représentation graphique et la manipulation d'objets. Pour se rapprocher de la représentation mentale de l'utilisateur, nous avons essayé de séparer, autant que possible, la notion sémantique de l'objet de sa représentation topologique, qui est la structure de données permettant de le visualiser à l'écran. La manipulation des objets se fait grâce à des méthodes qui ont été définies dans le paragraphe IV-2.2.1.

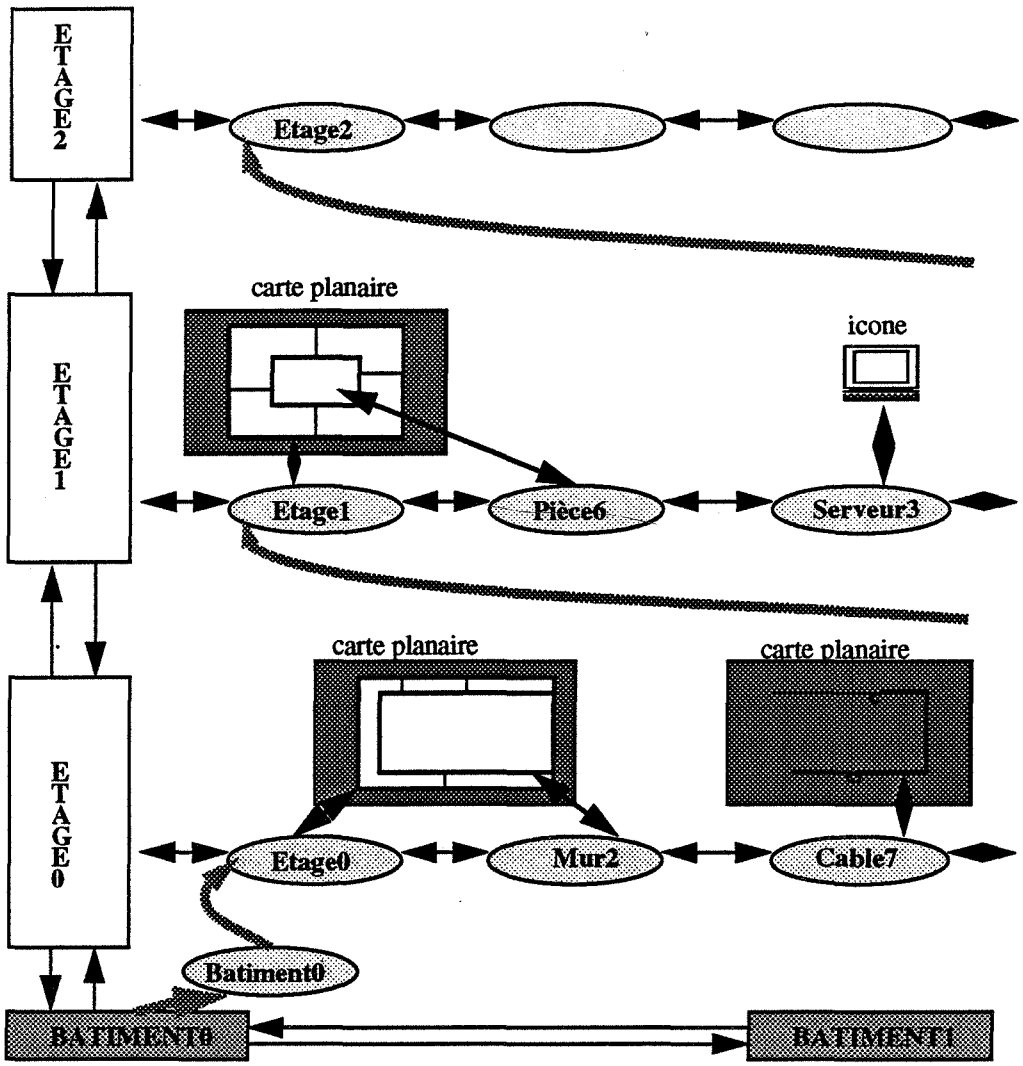


Figure 56: Les structures de données

A chaque étage, on a associé deux cartes planaires: une carte planaire (CP1) qui représente la topologie des murs et des pièces de l'étage et une carte planaire (CP2) qui représente la topologie des câbles du réseau de l'étage. Chaque arête de CP1 est la représentation d'un objet "mur" et chaque bord interne de CP1 est la représentation d'une "pièce". A l'objet "étage" est associé le plus grand bord externe de CP1. Un objet câble est composé d'un ou plusieurs bouts de câble. A chaque bout de câble, on associe une arête de CP2. On remarquera que dans la carte planaire CP2, la notion de bord (CP_globale) n'a pas d'importance.

Objets	Structure de données associée	Methodes
bâtiment	nœud d'une liste	Création, Destruction,Traduction
étage	le plus grand bord externe de CP1	Création, Destruction,Traduction
département	(nil)	Création, Destruction,Traduction
pièce	bord interne de CP1	Création, Destruction, Sélection, Affichage(2D,3D),Reclassification, Traduction
mur	arête de CP1	Création, Destruction, Sélection, Affichage(2D,3D), Reclassification, Traduction
gaine technique	liste de points	Création, Destruction, Sélection, Affichage(2D,3D), Traduction
faux plafond	(nil)	Création, Destruction, Sélection, Affichage(2D,3D), Traduction
faux plancher	(nil)	Création, Destruction, Sélection, Affichage(2D,3D), Traduction
gaine verticale (escalier, ascenseur, puit)	icône	Création, Destruction, Sélection, Affichage(2D,3D), Déplacement,Reclassification,Traduction
réseau	(nil)	Création, Destruction,Traduction
Câble	(nil)	Création, Destruction, Sélection, Reclassification,Traduction
bout de câble	arête de CP2	Création, Destruction,Sélection,connexion, Affichage(2D,3D)
câble vertical	arête	Création, Destruction,Sélection,connexion, Affichage(3D), Traduction
équipements (disques,stations,...)	icônes	Création, Destruction,Sélection,connexion, Affichage(2D,3D),Reclassification, Déplacement, Traduction

Figure 57: Tableau récapitulatif

L'avantage de la structure de données adoptée séparant l'objet de sa représentation est la flexibilité. En effet, on peut rajouter des objets qui ont une représentation graphique différente de celles qui existent. Il suffit de définir les nouvelles structures de données et les méthodes d'affichage pour chaque nouvel objet.

IV-3.3.1La Création des objets

La structure de données carte planaire a été définie dans le chapitre III. Cependant, dans notre module, nous avons besoin, pour chaque étage de deux cartes planaires: l'une représentant les murs et les pièce (CP1), l'autre représentant les câbles (CP2). La structure de données manipulée est donc la suivante:

```
Structure carte{
entier indice;          /*indice dans un tableau indiquant la carte active */
structure CP_carte *BN_carte[2];
structure Objet *objets;
};

Structure Objets{
entier type_objet;
entier identificateur;
entier x,y;
union  {
        structure Arête *arête;
        structure Bord *bord;
        structure Point *list_de_points;
        structure icone *image_obj;
        };
structure Lien *connexion;
structure Objet *obj_suivant,*obj_avant;
};

Structure CP_carte{
Environnement environ;          /* les structures Sommet, Point et
structure Point * list_de_points;        carte_planaire ont été définies
structure Sommet *sommet;                dans le chapitre III */
structure carte_planaire *bord;
};
```

Lorsqu'on active l'item "Contruct_PM" ou lorsqu'on modifie la carte planaire, l'algorithme suivant est activé:

```
début{
(1)     prétaitement(Liste_point);
        si (incrémental)
(2)             mise_à_jour_CP(Liste_point,CPi);
```

```
        sinon
(3)         construction_CP (CPi);
(4)     eliminer_arête_pendante(CPi);
(5)     créer_objet_CP(CPi);
        }
```

Les procédures [(1), (2), (3)] ont été discutées dans le chapitre III.

La procédure (4) parcourt la carte en recherchant les arêtes pendantes ayant une longueur supérieure à une valeur constante donnée et les supprime. Intuitivement, l'algorithme de cette procédure consiste à parcourir toutes les arêtes de la carte en marquant celles dont les deux brins appartiennent au même bord et ayant une longueur inférieure à une valeur donnée. Dans un deuxième parcours, elle élimine toutes les arêtes qui ont été marquées pendant le premier parcours. Cette procédure n'est appliquée qu'aux cartes planaires de type CP1.

La procédure (5) permet en quelques sorte d'extraire la sémantique contenue dans la carte planaire. En effet, elle parcourt l'arbre d'inclusion des bords et, pour chaque bord interne, elle crée un objet "pièce". De plus, pour chaque arête du bord, la procédure (5) crée un objet "mur". Dans le cas où la carte est de type CP2, pour chaque arête de la carte, elle crée un objet "bout de câble".

Un objet câble est, par définition, une succession d'objets de type "bout de câble". Pour créer un objet câble, on parcourt la suite des bouts de câble qui séparent deux connecteurs. Pour afficher le câble, on affichera tous les bouts de câble qui le constituent.

Quand l'utilisateur dessine avec la souris une gaine horizontale, l'OD crée une liste de points. Ces points ont été indiqués par l'utilisateur en appuyant sur le bouton gauche de la souris. Une gaine horizontale doit longer les murs des pièces qu'elle traverse, or les points fournis par l'utilisateur sont approximatifs. Il est donc nécessaire de les recaler de façon à ce qu'ils coïncident avec les murs. L'algorithme de recalage de la gaine horizontale consiste à parcourir la liste des points et, pour chaque point, chercher l'arête la plus proche, puis à projeter perpendiculairement ce point sur cette arête à une distance constante (voir la figure 57)

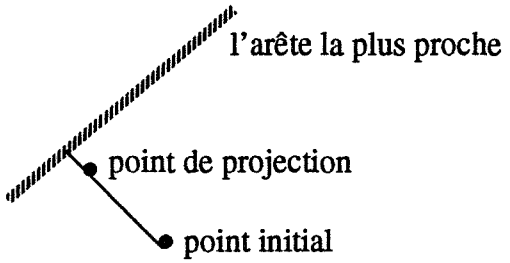


Figure 58: Recalage de la gaine horizontale

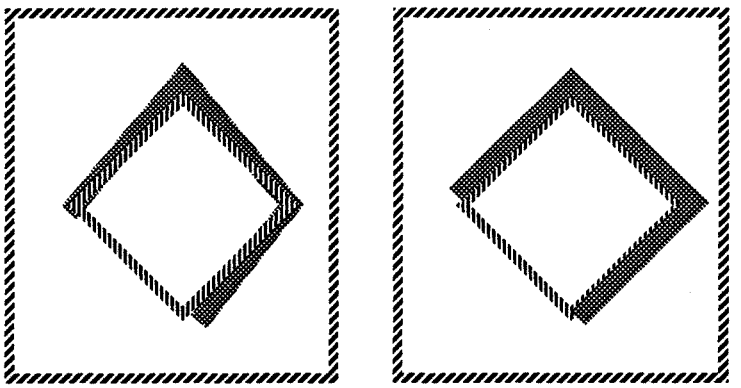


Figure 59 : Création d'une gaine horizontale

Pour tous les objets qui ne sont pas représentés dans la carte planaire, la méthode crée l'objet et lui associe sa représentation graphique, s'il en a une. Par exemple, l'objet station de travail a comme représentation graphique une icône; on associe donc à cet objet une structure de données support d'une icône.

IV-3.3.2 La destruction des objets

Cette méthode détruit l'objet et sa représentation. Dans le cas où l'objet est associé à une structure de la carte planaire, la destruction de l'objet provoque la mise à jour de la carte planaire. On notera, par exemple, que la destruction de l'objet mur peut provoquer parfois, après mise à jour de la carte planaire, la destruction d'un objet de type pièce.

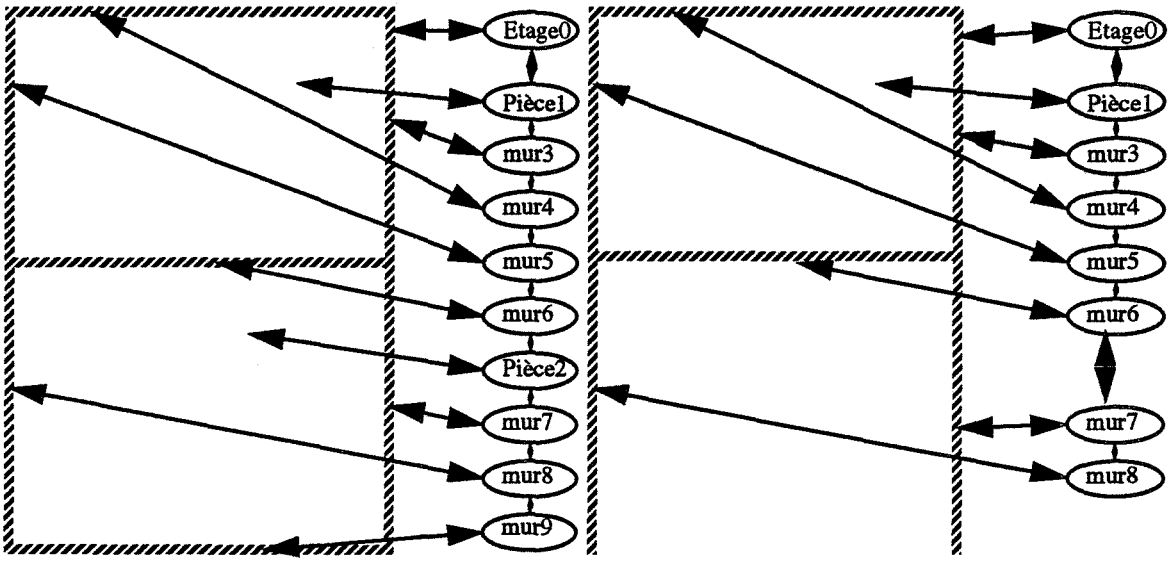


Figure 60: Destruction d'une arête

IV-3.3.3 La sélection d'un objet

Etant données les coordonnées (x,y) du point sélectionné, la méthode de sélection fonctionne selon le type de l'objet. Si la sélection concerne les objets du bâtiment, on commencera la recherche par les objets qui ont une représentation icônique; si on ne trouve pas l'objet, on continuera la recherche parmi les objets murs, et enfin on recherchera l'objet parmi les pièces. Si la sélection concerne les objets du réseau, on commencera la recherche par les objets qui ont une représentation icônique; si on ne trouve pas l'objet, on recherchera les objets "bout de câble".

- Recherche des objets qui ont une représentation icônique: dans ce cas, la méthode de sélection consiste à rechercher l'objet dont la boite englobante contient le point (x,y).

- Recherche des pièces: la méthode de sélection effectue un parcours de l'arbre d'inclusion en recherchant le plus petit bord interne contenant le point (x,y). L'algorithme est le suivant:

```
Bord *trouver_le bord (bord ,X,Y)
/* on parcourt l'arbre à partir de la racine qui est le bord englobant tous les
bords; si l'objet est dans un bord, on recherche parmi ses fils celui qui contient
l'objet jusqu'à obtenir le plus petit bord contenant cet objet; sinon on recherche
parmi les frères de ce bord. Si on ne trouve pas parmi les frères, alors ce bord est
le plus petit bord contenant cet objet
*/
{
/* X, Y coordonnées de l'objet */
bord ←racine le l'arbre;
Le_bord ←NULL
si <dans_ce_bord( bord, X,Y)> alors
        Le_bord← trouver_le_bord(bord->fils_ainé,X,Y);
sinon
        Le_bord← trouver_le_bord(bord->frère,X,Y);
si (Le_bord==NULL) alors Le_bord← bord;
retourner(Le_bord);
}

entier dans_ce_bord (bord,X,Y)
/* Le principe de cette procédure consiste à tracer une demi-droite aléatoire
partant du point (X,Y), puis à calculer le nombre d'intersections de cette
demi-droite avec les arêtes du bord. Si ce nombre est impair, alors le point est
situé à l'intérieur du bord, sinon il est à l'extérieur. Si le rayon aléatoire passe par
un sommet ou si il se superpose à un segment du bord, alors on relance un autre
rayon aléatoire. */
```

```
{
arrêt ←FAUX;
nb_intersection←0;
tanque (stop==FAUX) faire
{
demi-droite ← droite_aléatoire(x,y);
arête← bord->départ;
 faire {
        S← intersection( demi-droite, arête);
        si (S) et (S égale à un sommet du l'arête) alors stop← vrai
        sinon
                {nb_intersection++;
                arête← <arête suivante du bord>
        } tanque ((arête!=bord->départ) &&(stop==FAUX);
si (stop==VRAI) alors
                {
                 arrêt← FAUX;
                 nb_intersection← 0;
                }
sinon stop←VRAI;
}
si (nb_intersection est impair) alors <le point est à l'intérieur du bord>
sinon <le point est à l'extérieur du bord>
}
```

•recherche d'une arête: la méthode de sélection recherche parmi les objets qui ont pour représentation graphique une arête de la carte planaire, celle dont la distance avec le point (x,y) est inférieure à une valeur constante donnée. L'algorithme est le suivant

```
recherche_arête (x,y);
{
Objet_trouvé ← nil;
objet ← <premier objet de la liste des objets>;
dist ← DISTANCE          /* DISTANCE est la valeur constante*/
tanque ((objet !=nil) et (objet->type==ARETE))
        {
        arête ← objet->représentation.arête;
        (x1,y1) ← <projection perpendiculaire de (x,y) sur arête>
        d ← distance((x,y),(x1,y1));
        si d<dist alors
                        {
                        Object_trouvé ← objet;
                        d ← dist;
                        }
        objet ←objet->obj_suivant;
```

```
        }
retourner(Objet_trouvé);
}
```

IV-3.3.4 L'affichage des objets

Pour permettre à l'utilisateur de saisir des tracés dont la taille dépasse celle de la zone de dessin, on a utilisé une "bitmap" de grande taille (1500 pixels x 1500 pixels). La zone de dessin est alors une fenêtre sur la "bitmap" qui se déplace à l'aide de deux ascenseurs, l'un vertical et l'autre horizontal. La procédure d'affichage opère sur la "bitmap" et affiche seulement la zone visible sur la fenêtre. Dans sunview, le repère de la "bitmap" et des fenêtres est situé en haut à gauche, alors que le repère de l'utilisateur est situé en bas à gauche.

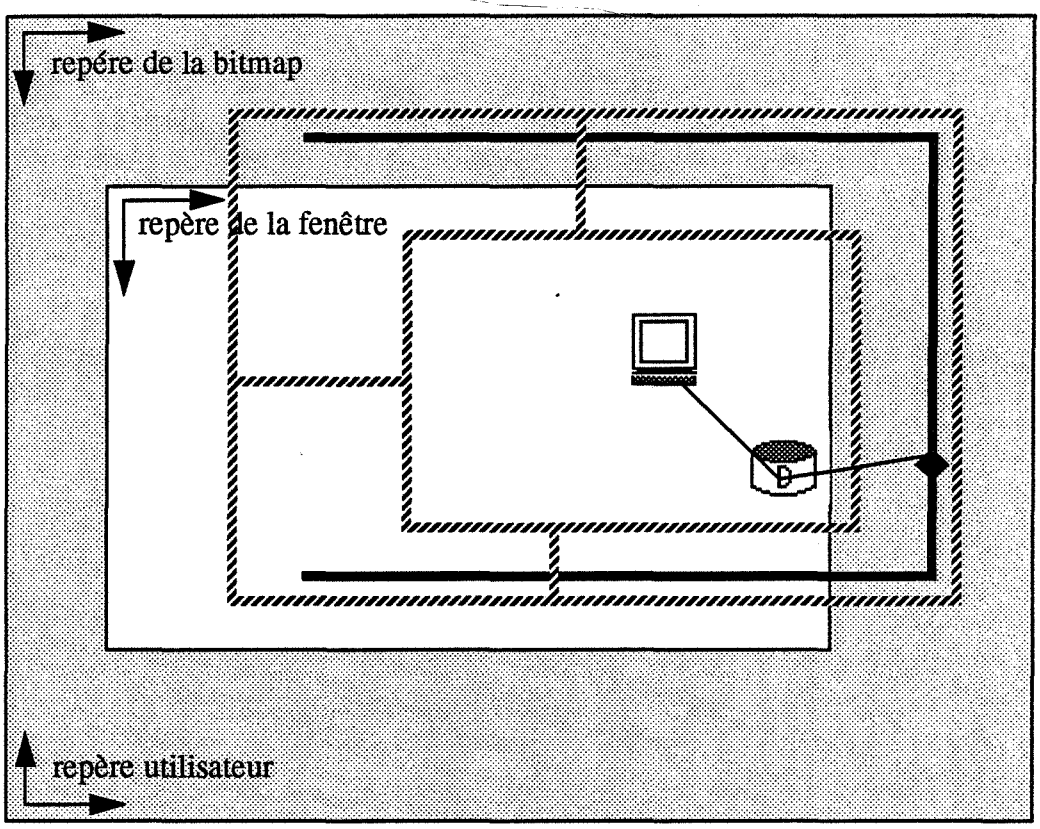


Figure 61: Les Bitmaps

IV-3.3.5 La connexion des objets

Dans la conception de réseaux, on a souvent besoin de connecter les équipements du réseau. Connecter deux objets revient à créer un lien entre eux. Un lien est constitué de deux connexions qui pointent sur les deux objets à connecter. De plus, un objet peut être connecté plusieurs fois et donc, il peut avoir une liste de liens. La structure de données

utilisée est la suivante:

```
structure connexion{
        structure objet *obj;
        structure Liens lien_suivant;
        }

structure Liens{
        entier type_lien;
        entier indice;
        structure connexion tab_connect[2];
        }
```

Beatrice Cahour [Caho 89] a montré que, généralement, les utilisateurs représentent les connexions par des segments de droite joignant les deux objets. Seules les connexions avec le câble sont représentées par une projection de l'objet sur le câble.

Nous utilisons cette structure de données, non seulement pour décrire les connexions entre objets, mais aussi, pour décrire toute relation liant deux objets; par exemple, la relation "dedans" qui décrit le fait qu'un objet se trouve à l'intérieur d'une pièce est représentée par un lien entre l'objet est la pièce.

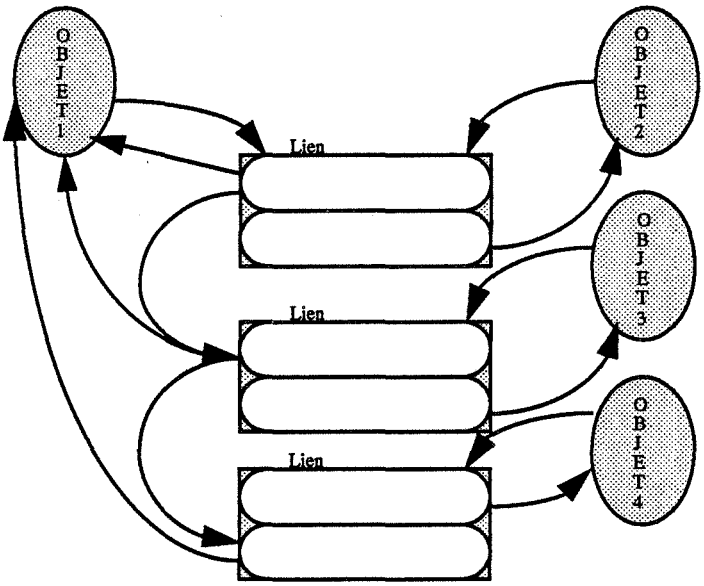


Figure 62: Les liens entre les objets

IV-3.3.6 La reclassification

L'opération de reclassification permet à l'utilisateur ou au système de reclassifier

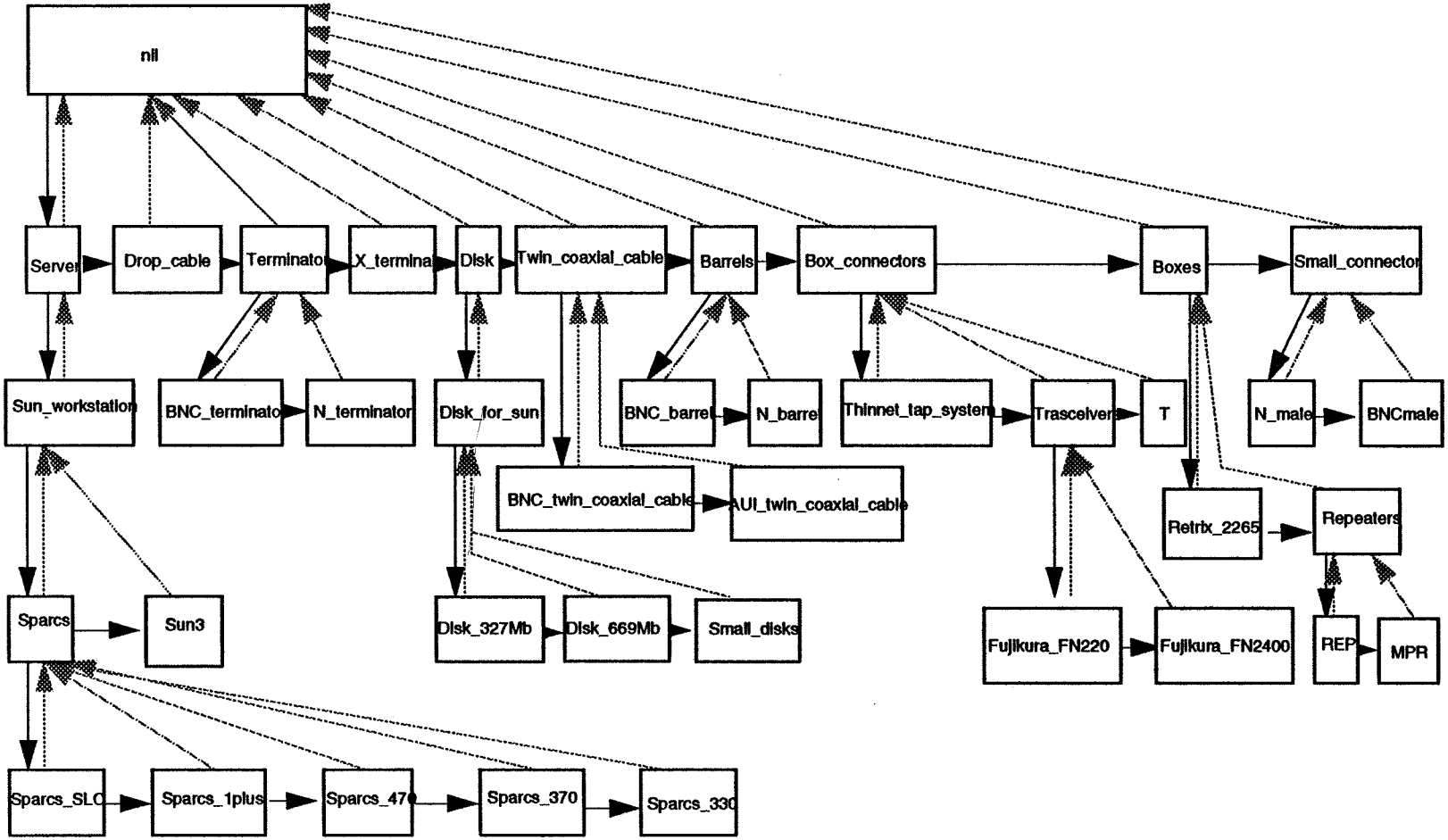


Figure 63: Arbre des icônes

un objet, c'est-à-dire de raffiner ou de généraliser la définition d'un objet. Remarquons que les objets manipulés dans l'OD sont représentés dans l'arbre sémantique de l'expert sémantique. Toutefois, pour pouvoir exploiter la relation (classe, sous-classe) définie par cette aborescence, les objets dans l'OD sont représentés par un arbre. La hiérarchie de cet arbre est inspirée de l'arbre sémantique en éliminant tous les objets qui ne peuvent être désignés graphiquement.

L'opération de reclassification n'est autre qu'un déplacement dans cet arbre. La figure 63 représente une partie de l'arbre utilisé par l'OD pour reclassifier les objets.

IV-4 DEDUCTION DE LA SEMANTIQUE

La déduction de la sémantique se traduit par la mise à jour de la base de connaissances de l'application à partir des structures de données des cartes planaires. Les informations extraites des structures de données sont décrites ci-dessous:

bâtiment ***(CLASS Buildings)***
Centre */* coordonnées du point situé au centre du premier étage*/*

étage ***(CLASS Floors)***
set_in_building */* identificateur du bâtiment contenant l'étage */*
height */* hauteur de l'étage, par défaut: 300*/*

département ***(CLASS Department)***
set_in_building

pièce ***(CLASS Locations)***
set_in_building */*identificateur du bâtiment contenant la pièce */*
set_in_floor */*identificateur de l'étage où se trouve la pièce */*
set_in_departement
orign_coordinates */* coordonnées du premier sommet du bord représentant la pièce*/*
walls_succession */* liste des murs de la pièce*/*

mur ***(CLASS Walls)***
location */*liste des identificateurs des pièces auxquelles appartient le mur*/*
angle
length

gaine ***(CLASS Horizontal_shafts)***
set_on_walls */* liste des murs le long desquels passe la gaine*/*

faux plafond ***(CLASS False_ceil)***

locations */*identificateur de la pièce*/*

faux planché*(CLASS False_floor)*
locations */*identificateur de la pièce*/*

gaine verticale*(CLASS Stair, Lift ,Verical_shaft)*
set_in_rooms */*liste des identificateurs des pièces traversées par la gaine*/*
joigned_floors */*liste des identificateurs des étages traversés par la gaine*/*
coord_x_y */*position de la gaine par rapport à la pièce de l'étage le plus bas*/*

réseau *(CLASS Networks)*

cable *(CLASS Thin_cables,Thick_cables,Cables)*
location
connected_elements */*la liste des objets connectés*/*
length
pass_in */* gaine dans laquelle passe le câble*/*
part_of */*réseau auquel appartient le câble*/*

équipement *(CLASS /* tous les objets qui ont une représentation icônique*/)*
location
connected_elements
part_of

Les deux prédicats suivants suffisent pour transférer les informations extraites du gestionnaire graphique. Celui-ci transforme ces informations en expressions CMR et les envoie au contrôleur de dialogue.

putobjects/3
/ ce prédicat permet de créer des instances d'une classe donnée*/*
arg1: identificateur de l'objet
arg2: IN
arg3: la classe de l'objet

putobjectvalue/3
/ ce prédicat permet de modifier la valeur des attributs de l'objet*/*
arg1: identificateur de l'objet
arg2: nom de l'attribut
arg3: valeur de l'attribut; */*cette valeur peut être un atome ou une liste d'atomes*/*

Le principe de la déduction peut être décrit de façon informelle ainsi: parcourir la liste des objets de tous les étages, et pour chaque objet activer la méthode de traduction qui

lui est associée. Les méthodes de traduction collectent les informations à partir des structures de données de l'OD, selon les définitions précédentes et les envoie au gestionnaire graphique en utilisant les prédicats "putobjects/3" et "putobjectvalue/3".

IV-5 COMMUNICATION ENTRE L'OD ET LE GESTIONNAIRE GRAPHIQUE

L'architecture de l'expert graphique montre que la communication entre l'OD et le reste du système passe nécessairement par le gestionnaire graphique. En effet, le gestionnaire graphique sert d'intermédiaire entre l'OD et le reste du système. Il transforme les informations reçues de l'OD en expressions CMR et inversement, il transforme les expressions CMR fournies par les différents experts du système en prédicats Prolog qui permettent d'activer des méthodes (procédures) de l'OD.

La communication entre l'OD et le gestionnaire graphique est présentée ici selon deux aspects: en mode entrée, permettant à l'utilisateur de communiquer avec le système; en mode sortie, permettant au système de présenter les informations à l'utilisateur.

IV-5.1Mode entrée

En mode entrée, l'OD envoie les informations sous forme de prédicats Prolog. Nous avons présenté dans le paragraphe 4.3 la déduction sémantique qui décrit les plans saisis par l'utilisateur au système. De plus, le système est informé par l'OD des manipulations effectuées par l'utilisateur. En effet, chaque opération accomplie par l'utilisateur produit un prédicat qui informera le système et lui demandera l'autorisation d'exécuter cette opération. Si l'autorisation est refusée, l'OD informe l'utilisateur qu'il lui est impossible d'exécuter cette opération. Une description détaillée des prédicats utilisés est fournie dans l'annexe B.

IV-5.2 Mode sortie

Le gestionnaire graphique dialogue avec l'OD en utilisant des prédicats Prolog. Ces prédicats consistent à activer des procédure écrites en C; il ont la forme suivante:

```
extern ([a_C_func],['fichier.o']).
gr_prédicat(liste_variable):-
        a_C_func (liste_variable).
```

"a_C_func" est une procédure écrite en C et qui doit être déclarée préalablement comme étant une fonction externe.

Cependant, quand l'application propose une solution de câblage d'un bâtiment, elle doit pouvoir présenter graphiquement cette solution sous une forme graphique. L'application construit alors des expressions CMR qui décrivent la scène graphique; c'est

alors que le gestionnaire graphique transforme ces expressions en des appels à des fonctions permettant de manipuler l'outil de dessin.

Cette approche est efficace; le seul problème est dû à ce que l'application ne manipule pas les informations géométriques telles que la position exacte dans une pièce d'une station de travail ou les coordonnées des extrémités de chaque bout de câble. Sans ces informations géométriques, il est difficile à l'outil de dessin de représenter graphiquement la solution proposée par l'application. Cependant, les informations topologiques fournies par l'application peuvent être utiles pour l'outil de dessin et lui permettre de trouver une solution d'affichage du réseau. Nous proposons, ci-dessous, deux solutions à deux problèmes précis. Le premier est l'affichage d'un objet ayant une représentation icônique; le deuxième est l'affichage des câbles.

IV-5.2.1- Affichage d'un objet "icônique"

Lorsque l'application demande à l'expert graphique d'afficher un objet ayant une représentation icônique dans une pièce, il ne fournit pas de coordonnées précises. Une première solution consiste à générer des coordonnées aléatoires dans la pièce. Cette solution présente des inconvénients. En effet, l'emplacement des objets étant aléatoire, les icônes et les connexions peuvent se superposer (voir la figure 64).

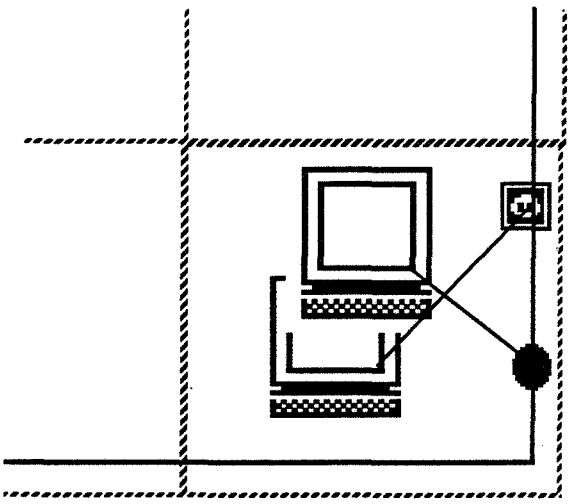


Figure 64: Superposition des icônes

Une autre solution possible, qui pourrait être implémentée prochainement serait de tenir compte des connexions entre les objets et d'utiliser une grille d'affichage. L'algorithme permettra d'afficher les objets connectés l'un à coté de l'autre, tout en tenant compte de l'espace libre de la pièce. L'espace de la pièce est maintenu à jour par une grille, plus au moins fine, qui indique les zones non occupées de la pièce.

IV-5.2.2- Affichage des câbles

L'affichage des câbles est un problème délicat à résoudre. En effet, les informations fournies par l'application sont insuffisantes et ne décrivent pas la géométrie des câbles. A partir des informations topologiques, l'outil de dessin doit extraire des notions géométriques lui permettant d'afficher les câbles. Cependant, il faut imposer des contraintes pour résoudre efficacement le problème. Ces contraintes sont:

- Les extrémités du réseau sont connectées à des terminateurs.
- Les câbles longent nécessairement les murs.
- Un câble est connecté par ses extrémités à des objets

Pour chaque câble l'application fournit les informations suivantes:

- La liste ordonnée les murs longés par le câble;
- La liste des pièces traversées par le câble. On remarque que la liste des pièces peut être redondante; cela permettra de déterminer de quel côté du mur se trouve le câble.
- La longueur du câble.

A partir de ces informations, l'OD doit dessiner le câble de manière à ce qu'il longe les murs. L'algorithme d'affichage comprend deux étapes: la première consiste à parcourir tous les câbles, en partant d'un terminateur et en utilisant la connectivité des câbles avec les objets, pour marquer les murs qu'ils longent et déterminer la position approximative des extrémités du câble. Cette position est déterminée en parcourant les bords dans le sens des aiguilles d'une montre; ainsi les câbles seront totalement ordonnés sur chaque mur. La deuxième étape consiste à parcourir encore une fois les câbles, en connaissant le nombre exact de câbles qui longent le mur, leur ordre sur celui-ci et leur longueur; l'algorithme déterminé alors la position exacte de chaque câble. On peut formuler la deuxième étape de l'algorithme comme suit:

```
AFFICHE_CABLE
    {
    O ← un terminateur
    faire {
        C← le câble, non marqué, connecté à O;
        M← le mur longé par C;
        p0← position du câble sur le mur /*cette position a été déterminée pendant
                                          la première étape. La position du
                                          câble est définie en parcourant les
                                          bords dans le sens des aiguilles d'une
                                          montre */
        tanque liste des murs ≠ []
        faire {

            si (p0 est la plus à droite) alors
```

```
                p1←le sommet non marqué de M
            sinon p1← la position de l'autre extrémité du câble;
            créer un bout de câble (p0,p1);
            placer O en p0;
            M← mur suivant;
            p0← p1;
            }
        marquer C,
        marquer O;
        O← l'objet suivant connectés à C
        } tanque O ≠ terminateur;
}
```

Cet algorithme assez simple donne un bon résultat si on respecte les contraintes définies précédemment. Cependant, l'algorithme ne peut pas afficher des cables qui se superposent. La figure 65 donne quelques exemples des possibilité de cet algorithme:

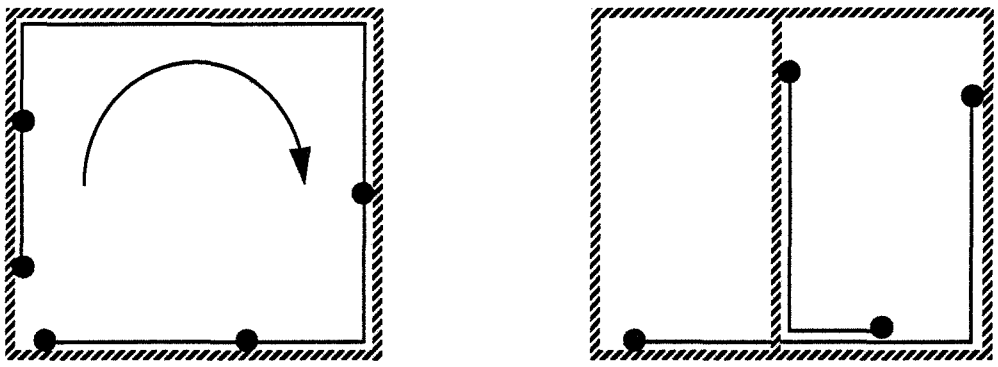


Figure 65: Affichage des câbles

IV-6 PORTABILITE DE L'OD

IV-6.1 Dépendance vis-à-vis de Sunview

L'OD utilise le système de fenêtrage Sunview, mais il est possible d'adapter le module pour l'utilisation d'un autre système de fenêtrage tel que X-window. La figure 66 montre la dépendance de l'OD vis-à-vis du système de fenêtrage: seules les parties colorées dépendent de Sunview. En effet, nous avons développé une bibliothèque graphique qui utilise des fonctions Sunview de bas niveau. Les fonctions développées peuvent être classées en quatre groupes:

- les fonctions de traçage: ces fonctions permettent de tracer des lignes et des polygones;
- les fonctions de remplissage: ces fonctions permettent de remplir une forme géométrique avec une texture prédéfinie;
- les fonctions d'affichage: ces fonctions permettent d'afficher du texte et des "bitmaps";
- les fonctions de création/destruction: ces fonctions permettent de créer et de détruire des "bitmaps" et des fenêtres.

A partir de ces fonctions de bas niveau, nous avons développé d'autres fonctions qui permettent la création et la manipulation des menus, des boutons, des barres de défilement. Ces fonctions sont indépendantes de Sunview. Une adaptation de l'outil au système de fenêtrage X-window consistera alors, d'une part à redéfinir les fonctions de traçage, de remplissage, d'affichage et de création/destruction, d'autre part à redéfinir l'interface qui gère les événements en entrée du noyau de l'OD.

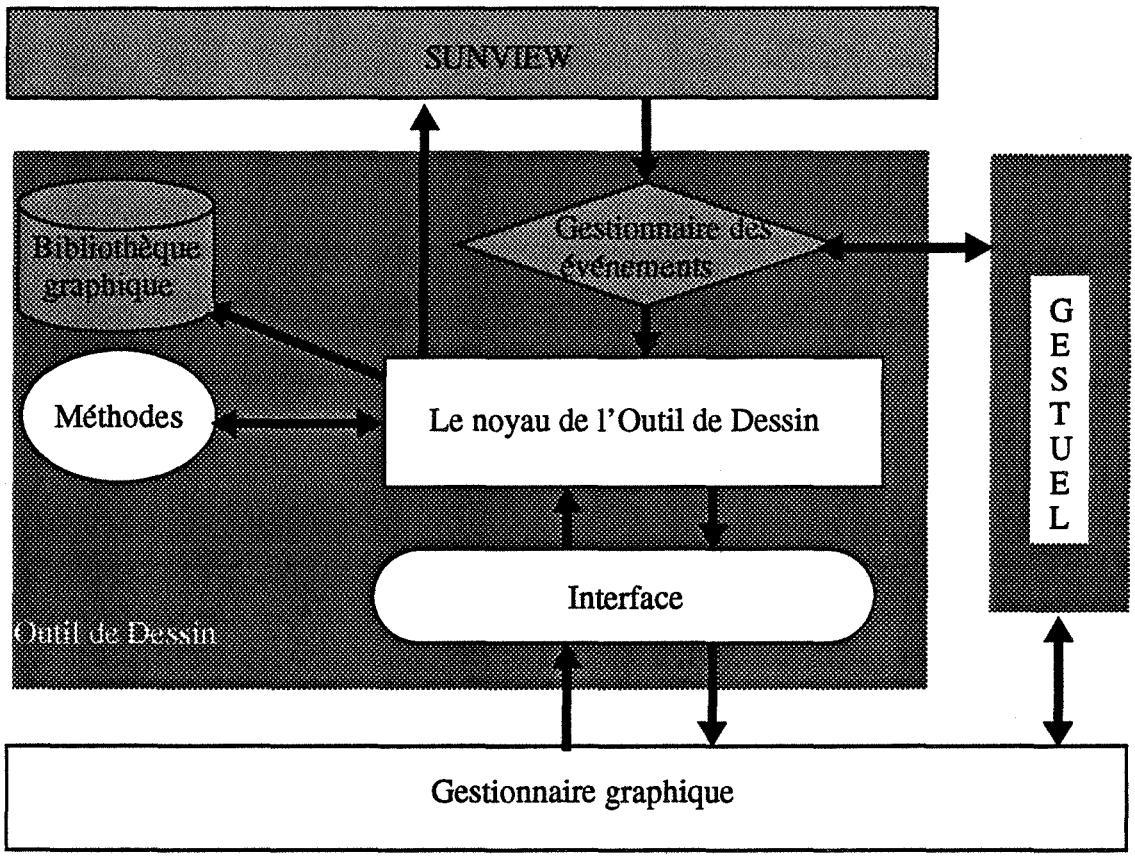


Figure 66: Dépendance vis-à-vis de Sunview

IV-6.1.1 Le modèle Sunview

Sunview est un système de fenêtrage orienté objets. L'idée de base de Sunview est l'utilisation d'objets visuels pour la construction d'une interface. Plusieurs types d'objets sont définis, chaque type d'objet ayant des propriétés particulières qui peuvent être utilisées pour une tâche précise.

La plus importante classe d'objets est la classe fenêtre "Windows". Il existe aussi d'autres classes d'objets visuels tels que les curseurs, les icônes, les menus et les barres de défilement. Pour manipuler ces objets, il faut fournir à une fonction associée l'identificateur unique de l'objet.

La figure 67 illustre les différentes classes d'objets visuels de Sunview:

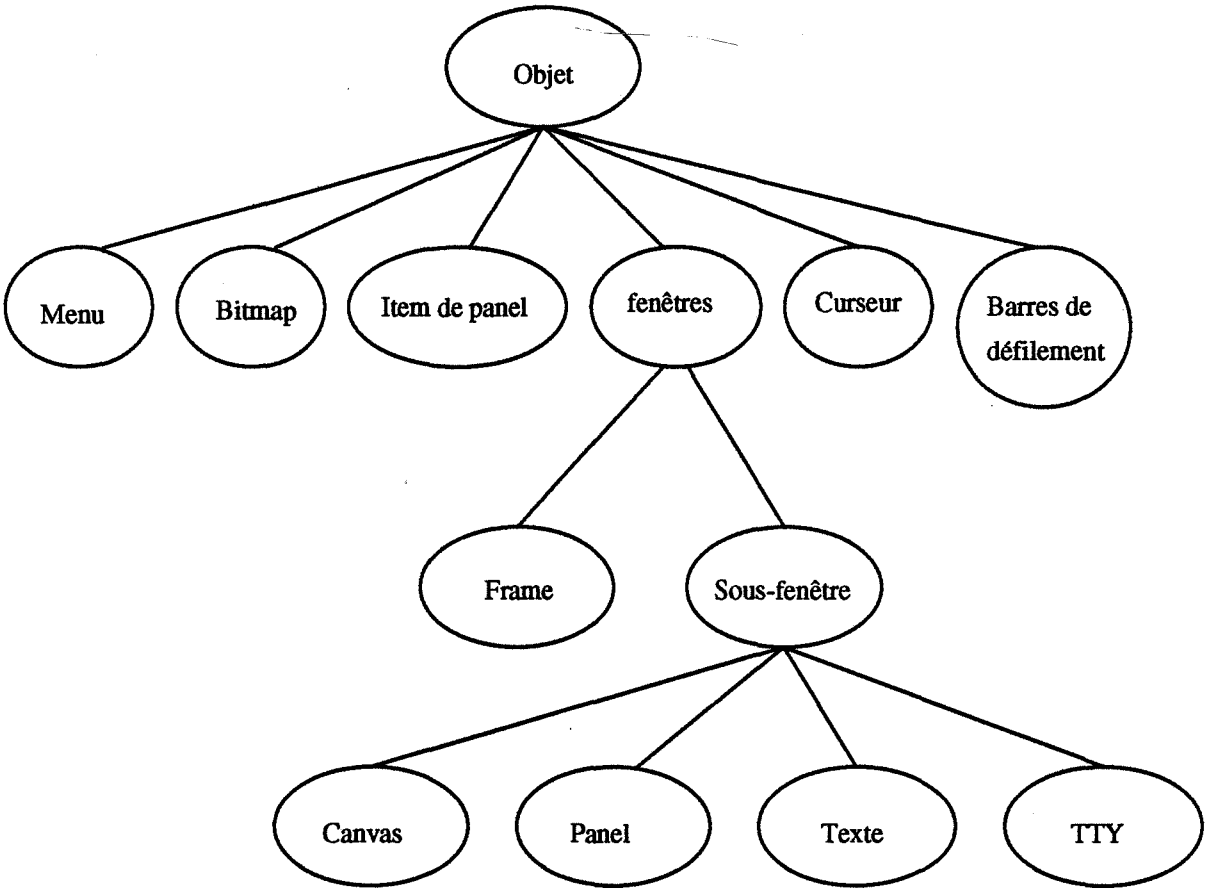


Figure 67: Le modèle de Sunview

IV-6.1.2 Le Modèle de l'OD

Le modèle de Sunview n'est pas très flexible dans le sens où l'utilisateur doit définir l'interface en utilisant les objets prédéfinis. Pour cette raison et pour être le plus indépendant possible de Sunview, nous avons défini un modèle pour l'OD, inspiré du modèle de Sunview. Nous avons redéfini les classes Menu, Icône, Item, et Barres de défilement en les adaptant aux besoins de l'application et en créant plusieurs styles pour chaque objet: par exemple, l'utilisateur peut créer des menus fixes avec une barre de défilement (voir l'annexe B).

De plus, dans le modèle de l'OD, nous utilisons seulement les classes "Frame" et "Canvas" définies par Sunview. Un objet "Canvas" est composé de zones. Chaque zone est utilisée pour une tâche spécifique et possède une fonction qui décrit son comportement. Ainsi, l'utilisateur peut définir facilement un nouveau type de zone en spécifiant son aspect visuel et sa fonction de comportement (voir l'annexe B).

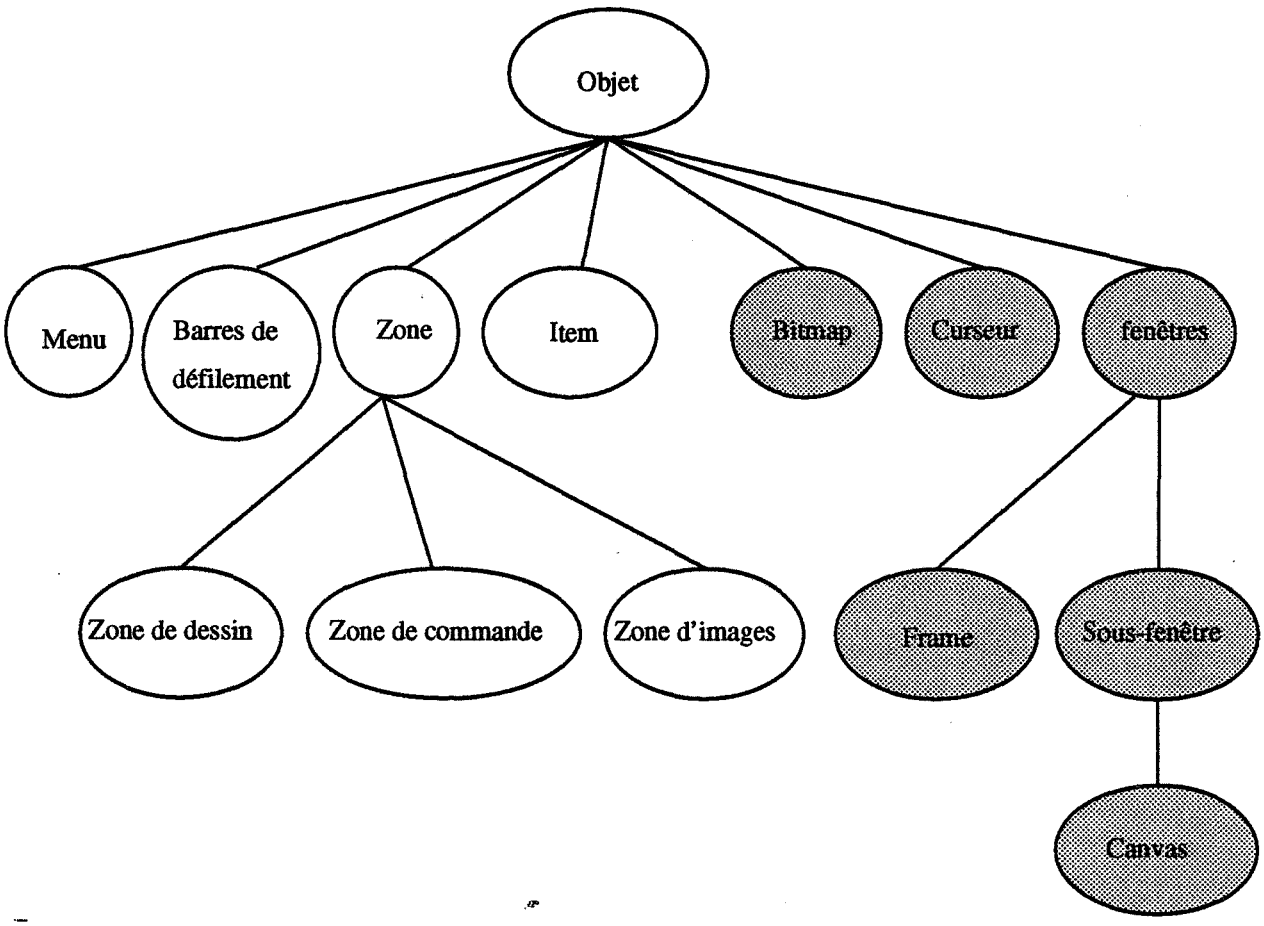


Figure 68 : Le modèle de l'OD

IV-6.2 Dépendance vis-à-vis de l'application

Les parties de l'OD qui dépendent de l'application sont essentiellement les méthodes appliquées aux objets. En effet, le comportement des objets et leurs utilisation diffèrent selon l'application. Par exemple, pour une autre application, les méthodes de traduction peuvent être différentes dans le sens où la sémantique extraite des structures de données peut ne pas être la même que pour MMI2.

Les icônes sont stockées dans une base de données sous forme de tableaux d'entiers. Si pour une nouvelle application, on a besoin d'autres icônes, il sera facile de les ajouter à la base de données.

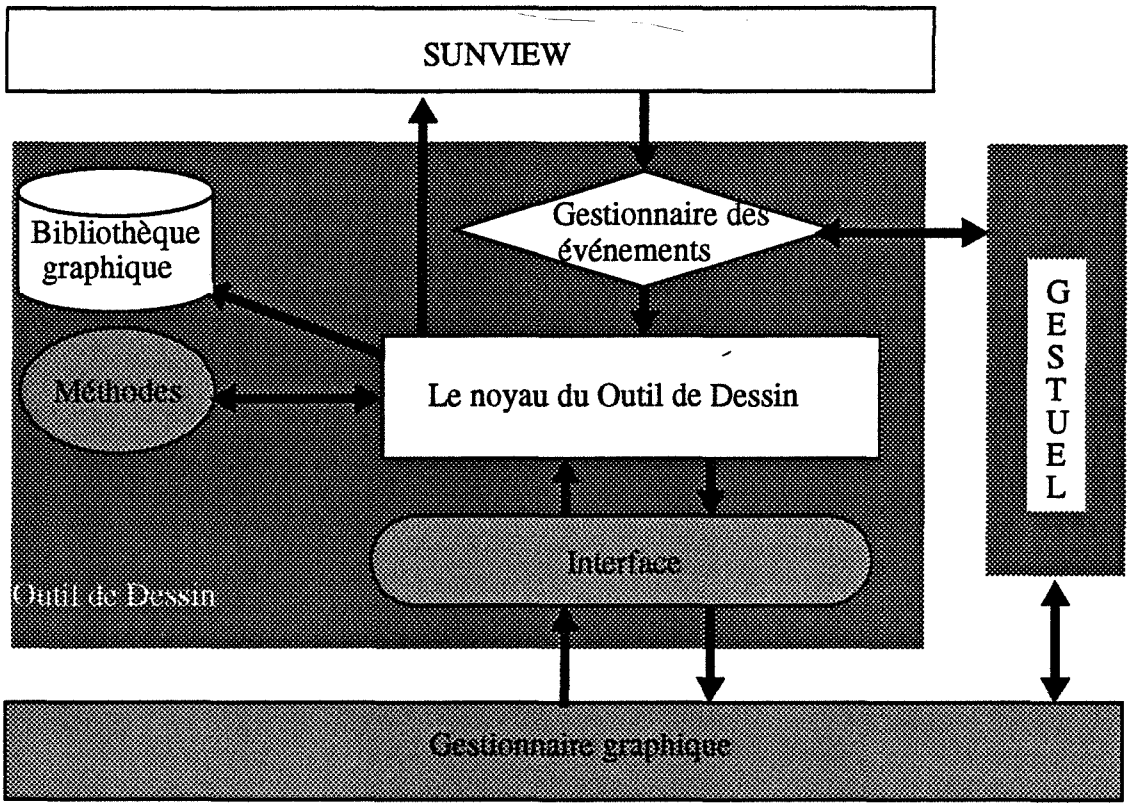


Figure 69: Dépendance vis-à-vis de l'application

Remarquons que l'OD est complètement indépendant de la CMR. En effet, tous les modules qui manipulent la CMR sont situés dans le gestionnaire graphique.

CONCLUSION

Modestement, cette thèse a proposé un outil de manipulation directe dans un environnement Multi-Modes qui s'inscrit dans le cadre d'un projet ESPRIT II baptisé MMI2. La durée globale du projet est de cinq ans; au bout de trois ans, nous avons produit un prototype intégrant les différents modules du système et présenté lors de la semaine ESPRIT91 à Bruxelles. Dans l'état actuel, le code de l'outil de dessin fait 22000 lignes écrites en C et en BimProbe.

L'approche utilisée pour le développement de l'outil de dessin est basée sur les constructions géométriques. L'utilisation des cartes planaires s'est avérée intéressante, surtout pour la représentation et l'affichage de scènes 2D. Par contre, pour des scènes 3D, les cartes planaires sont complètement inefficaces; une méthode de visualisation a été adoptée pour que l'utilisateur ait l'impression d'une vue 3D. Cependant, l'utilisation des 3-G-cartes [Lien 89] est une solution qui peut être retenue pour représenter des scènes 3D si les performances de l'algorithme sont satisfaisants dans un environnement interactif.

Nous avons constaté que l'imprécision de l'arithmétique flottante rend inutilisable l'algorithme de Bentley-Ottmann, du moins sous sa forme initiale. En effet, l'exploitation de la cohérence du plan est incompatible avec l'imprécision numérique. Le cas de la méthode de Bentley-Ottmann prouve qu'il ne suffit pas de démontrer la validité d'un algorithme géométrique pour que cet algorithme fonctionne correctement en machine. Il faut aussi démontrer que l'imprécision numérique ne nuit pas au bon déroulement de l'algorithme. Quand la méthode ne résiste pas à l'imprécision, elle doit utiliser une arithmétique parfaitement correcte (arithmétique rationnelle), dont le surcoût peut modifier les performances théoriques de l'algorithme.

Lors du développement de l'outil de dessin, nous nous sommes heurtés à un problème crucial posé par la séparation des aspects topologiques et géométriques. En effet, l'application, qui est une base de connaissances, ne fournit que des informations topologiques sur le réseau et le bâtiment; cependant, pour représenter graphiquement ces informations, l'outil de dessin a besoin d'informations géométriques sur les objets telles que leurs coordonnées par rapport à un repère quelconque. Une solution partielle a été proposée en imposant des contraintes. D'autre solutions peuvent être envisagées telles qu'un système expert qui permettra de gérer efficacement l'espace d'affichage et qui contiendra des connaissances nécessaires pour produire les informations géométrique à partir la topologie de la solution proposée par l'application.

D'une manière générale, certaines informations contenues dans l'application sont nécessaires au bon fonctionnement d'une interface intelligente, par exemple, quand l'utilisateur installe une machine à l'extérieur des murs du bâtiment l'interface de MMI2 est incapable d'interdire une telle action sans la consultation de l'application. De ce fait, deux solutions peuvent être envisagées: la première, nécessite l'interrogation de l'application à tout moment du dialogue entre l'interface et l'utilisateur. Cette solution à

été partiellement retenue par le système MMI2. Il est évident que cette approche affecte considérablement les performances du système par des accès à l'application qui sont parfois inutiles. La deuxième solution, se propose de dupliquer dans l'interface une partie des informations contenues dans l'application; ceci permettra des accès rapides à l'information quand c'est nécessaire. Cependant, il n'est pas facile d'extraire de l'application les informations utile pour l'interface car elles font partie d'un ensemble de connaissances qui infèrent les unes sur les autres.

Nous avons donné beaucoup d'importance à la flexibilité et à la réutilisabilité de l'outil de dessin en définissant des points d'attache précis et une structure de données assez générale ne dépendant pas de l'application. De plus une version utilisant X-window est envisagée dans les mois à venir. D'autres extensions que nous souhaiterions développer dans les années à venir sont:

- des extensions simples du module 3D: possibilité de création d'un étage de numéro quelconque (et pas seulement celui situé au dessus du dernier créé) et la possibilité de reprise d'une carte planaire d'un bâtiment ou d'un réseau pour décrire un nouvel étage,...

-une extension de la notion de "zoom", permettant de décrire un bâtiment important par une carte planaire des départements, chaque département étant associé à une autre carte planaire. Cette extension permettra bien entendu le passage d'une représentation à une autre simplement (et dans les deux sens);

-une extension des entités manipulées à celles de site comportant plusieurs immeubles géographiquement séparés;

- une évaluation des interfaces multi-modes à partir du projet MMI2 (notamment sur le choix des modes par l'utilisateur).

Nous ne pouvons conclure sans souligner l'apport d'un tel projet Européen où la participation et les échanges des idées entre les différentes équipes ont permis la réalisation de ce projet dans de bonnes conditions. L'intégration des différents modules du système été une tâche difficile et a exigé la contribution de tous les participants, malgré la difficulté de certains (!!) à s'exprimer en anglais. Il faut aussi souligner l'importance de la messagerie dans la réalisation de tel projet, en effet, sans le nombre incalculable de messages échangés (en anglais), il aurait été difficile, sinon impossible, de progresser.

BIBLIOGRAPHIE

[Allg 89] J. Allgayer, K. Harbush, A. Kobsa, C. Reddig, N. Reithinger & D. Schmauks, "XTRA: a Natural Language Access System to Expert Systems", International Journal of Man Machine Studies, 1989.

[Alt 87] J.L. Alt & G. Weir , "Dialogue design for dynamic systems", ESPRIT'87, Elsevier Science, Amsterdam, pp 819-825.

[Ama 90] H.B Amara & B. Peroche, "MMI2: Multi-Mode Interface for Man-Machine Interaction with Knowledge Based System", JISI'90, pp 214-223, Mai 1990.

[Ama 91] H.B. Amara., B. Peroche, M. Wilson & H. Chapel, "Graphical Interaction in a Multi-Modal Interface", ESPRIT'91 Conference Proceedings, pp 303-321, Novembre 1991.

[Ayal 85] D. Ayala, P. Brunet, R. Juan, I. Navaza, "Object Representing by Means of Nomina; division quadtrees and octrees", ACM Transaction on graphics, vol.4, n° 1, pp 41-59, January 1185

[Bass 88] L. Bass, E. Hardy, K. Hoyr, R. Little & R. Seacord, "The Serpent ru time architecture and dialogue model", Carnegie Mellon Technical Repport, CMU/SEI-88-TR-6, Junuary 1988.

[Baum 75] B. G. Baumgart, "A Polyhedron Representation for Computer Vision", In AFIPS conference, pp 589-596, 1975.

[Bela 82] A. Belaid, G. Masini, "Segmentation de Tracé Manuscrits sur Tablette Graphique en vue de leur reconnaissance", TSI, vol. 1, n° 2, pp155-168, 1982.

[Benn 86] J. L. Bennett, " Tools for Building advanced user interfaces", IBM System Journal, vol. 25, NOS 3/4, 1986

[Bent 79] J.L. Bentley, T. A. Ottmann, "Algorithms for reporting counting geometric intersections", IEEE, Trasaction on conputer, vol 28, n° 9, 1979.

[Bert 79] M. Berthod, P. Jancenne," Le Pré-traitement des Tracés Manuscrits sur Tablette Graphique", AFCET meeting, Reconnaissance des formes et intelligence artificielle, Toulouse, pp 195-209, September 1979.

[Bett 87] B. Betts, D. Burlingame, G. Fischer, J. Foley, M. Green, D. Kasik, S.T. Kerr, D.Olsen & J.Thomas, " Goals and Objectives for User Interface Software", Computer Graphics, Vol 21, N: 2, Avril 1987.

[Bino 89] J.L. Binot, "Overall System Architecture", ESPRIT P2474 MMI2, Deliverable d1, Juillet 89.

[Cael 91] J. Caelen, "Interaction Multimodale dans IPCdraw: expérience et perspective", PRC communication homme-machine, GRECO-PRC, lyon Avril 1991.

[Caho 89] B. Cahour, "Simulation of Interactions Between Users and KBS", Esprit P2474 MMI2, Volume 1, Juillet 1989.

[Caho 91] B. Cahour & Helen Chappel, "User Modelling for Multi-Modal Co-operative Dialogue with KBS", Esprit P2474 MMI2, Deliverable d3, Mai 1991.

[Call 87] L. A. Call, D. L. Cohrs, B. L. Miller, " CLAM: an Open System for Graphical User Interfaces", OOPSLA'87 Proceedings, pp 277-286, Octobre 1987.

[Chap 91] H. Chappel, M. Wilson, "The graphical presentation of structured representation", ESPRIT P2474 MMI2, Deliverable d38, Juin 1991.

[Cout 86] J. Coutaz, "La Construction d'Interfaces Homme-Machine", Rapport IMAG RR 635-I, Novembre 1986.

[Cout 88] J. Coutaz, " Interface Homme-Ordinateur, Conception et réalisation", Thèse de doctorat d'état , Université de Joseph Fourier de Grenoble, Décembre 1988.

[Cout 91] J. Coutaz & A. Gourdol, "Communication Homme-Machine Multimodale: Perspectives pour la Recherche", PRC communication homme-machine, GRECO-PRC, lyon Avril 1991.

[Cori 75] R.CORI, "Un Code pour les Graphes Planaires et ses Applications", Astérisque n° 27, 1975.

[Cram 87] C. Crampton, "MUSK: a Multi-user Sketch program", Proceedings of the European UNIX Systems User Group Conference, Dublin, september 1987.

[Dars 89] F. Darses, "Graphical Representations in Network Design", Rapport de Recherche, INRIA Avril 1989.

[Edmo 60] J.Edmonds, "A Combinatorial Representation for Polyhedral Surfaces", Notices of AMC, 7, 1960.

[Falz 86] P. Falzon, "Les dialogues de diagnostic: l'évaluation des connaissances de l'interlocuteur", Rapport de recherche n° 747, INRIA, 1986.

[Falz 89] P.Falzon & F. Darses, "The design activity in Networking, General remarks and first observations", MMI2 code INRIA/1, 1989.

[Fini 86] T. Fini & D. Drager, "GUMS: A General User Modeling System", Technical Repport MS-CIS-85-35, Department of Computer and Information Science, University of Pennsylvania, 1986.

[Fole 86] J.D. Foley & J.L. Sibert, "How To Design User-Computer Interfaces", ACM SIGGRAPH 86, Course n°18, 1986.

[Fole 87] J.D. Foley, "Transformations on a Formal Specification of User-Computer Interfaces", Computer Graphics., vol 21, n°2, Avril 1987.

[Gang 89] M.Gangnet, J.C. Herve, T. Pudet, J.M.V. Thong, "Incremental Computation of Planar Maps", Computer Graphics, vol. 23, n°3, July 1989.

[Gold 84] A. Goldberg, "Smaltalk-80, the Interactive Programming Environment", Addison-Wesley, 1984.

[Goul 88] J.D. Gould, "How to design Usable Systems", Handbook of Human-Computer Interaction, Elsevier Science, pp 757-789, 1988.

[Gree 86] M. Green, "A survey of three dialog Models", ACM transaction graphics, July 86.

[Gros 89] C. Gross, "Operations Topologiques et Géométriques sur les Multicartes Combinatoires, Doctorat de l'Université Louis Pasteur, Strasbourg, Avril 1989.

[Hans 88] S. Hansen , L. Holgard, M. Smith, "Eurohelp: Intelligent help systems for information processing systems", ESPRIT'88, pp 578-818.

[Haye 86] J.P Hayes, "Steps toward integrating natural language and graphical interaction for knowledge-based systems", Proceedings of the 7th European conference on Artificial Intelligence, pp 456-465, July 1986, Brighton.

[Hero 76] C. Herot, "Graphical Input through Machine recognition of Sketches", Computer Graphics, vol. 10, n° 2, pp 97-102, 1976.

[Hoff 89] C.M. Hoffman, J.E. Hopcroft, M.S. Karasick, "Towards Implementing Robust Geometric Computations", In proceedings of the Fourth Annual ACM Symposium on Computational Geometry, ACM Press, New York 1988.

[Holl 87] E. Hollnagel, G. Weir & G. Sundstrom, "User modeling in the Gradient Project", Esprit'87, pp 811-818 Elsevier Science, Amsterdam, 1987.

[Huds 87] S.E. Hudson, " UIMS Support For Direct Manipulation Interfaces", Computer Graphics, vol 21, n° 2, Avril 1987.

[Jacq 70] A. Jacques, "Constellation et Graphes Topologiques", In Combinatorial Theory and Applications, pp 251-260, Budapest 1970.

[Klei 87] E. Klein, " Dialogues with Language, Graphics and Logic", ESPRIT'87 Achievements and Impact, Part 1, Project n°393, pp 867-873, Septembre 1987.

[Lien 88] P. Lienhardt, "Extension of the Notion of Map and Subdivision of a Three-dimensional Space", In STACS'88, Lecture Notes in Computer Science 294, 1988.

[Lien 89] P.Lienhardt, "N-Dimentional Generalized Topological Maps", Rapport de recherche R 89-04, Université Louis Pasteur, Strasbourg, Mars 1989.

[Kris 89] R. Krishnamurti, "Representational semantics for graphical discourse", ACORD deliverable T3.6 Part I.

[Maso 88] J. Mason, J. Edwards," Surveying projects on intelligent dialogue", International Journal of Man-Machine studies, vol. 28, 1988, pp 259-307.

[Meye 89] B. Meyer, "Reusability: The case for object-oriented design", IEEE Software, pp 50-59, Mars 1989

[Mich 84] D. Michelucci & M. Gangnet, "Communication Homme-Machine et dessins des projets", Rapport final relatif à l'exécution d'un projet, Ecole Nationale Supérieur des Mines de Saint-Etienne, Septembre 1984.

[Mich 87] D. Michelucci, "Les Représentations par Frontières", Thèse Ecole Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, 1987.

[Mora 81] T. Moran " The Command language Grammar: a representation for the user interface of interactive computer systems", International Journal for Man Machine Studies, pp 3-50, 1981.

[More 90] J.M. Moreau, "Hiérarchisation et Facétisation de la Représentation par Segments d'un Graphe Planaire dans le Cadre d'une Arithmetique Mixte", Thèse Ecole des Mines de Saint-Etienne, Saint-Etienne 1990.

[Mull 78] D.E. Muller, P.F. Preparata, "Finding the Intersection of Two Convex Polyhedra", Theoretical Computer Science, vol. 7, n° 2, pp 217-236, October 1978.

[Myer 87] B.A. Myers, "Creating User interfaces by Demonstration", PhD Thesis, Department of Computer Science, University of Toronto, Technical Repport CSRI-196, May 1987.

[Nana 90] J. Nanard, " La Manipulation Directe en Interface Homme-Machine", Thèse d'état, Université Montpellier II, Décembre 1990.

[Over 81] M.H. Overmars, J.V. Leuwien, "Maintenance of Configurations in the Plane", Journal of Computer and System Science, vol. 3, pp 166-204, 1981.

[Olse 83] D.R. Olsen & E. P. Dempdey, " SYNGRAPH: a Graphical Interface Generator", Computer Graphics, vol. 17, n° 3, Juillet 1983.

[Pres 87] N. Preston, " Integrating Graphics into Prolog", ESPRIT'87 Achievements and Impact, Part 1, Project n° 973, pp 392-401, Septembre 1987.

[Rhy 87] J. Rhyne, R. Ehrich, J. Bennett, T, Hewett, J. Sibert & T. Bleser, "Tools and Methodology for User AInterface Development",Computer Graphics, vol 21,n° 2, Avril 1987.

[Scap 86] D.L.Scapin, "Guide Ergonomique des Interfaces Homme-Machine", Rapport technique, INRIA, Octrobre 1986.

[Sedl 89] D. Sedlock, "Common Meaning Representation", ESPRIT P2474 MMI2, Deliverable d2. Novembre 89.

[Sega 90] M. Segal, "Using Tolerances to Guarantee Valid Polygon reconstruction algorithm", Computer Graphics, vol. 24, n°4, pp 17-27, August 1990.

[Sibe 86] J. L. Sibert, W. D. Hurley & T. W. Bleser, " An Object-Oriented Interface Management System", ACM, Dallas, vol 20, n° 4, Aout 1986.

[Tutt 84] W.T. Tutte, "Graph Theory", Addison Wesley, 1984.

[Tyle 88] S. Tyler, " SAUCI: Aknowledge-Based Interface Architecture", Proceedings of CHI'88, pp 235-240, Addison-Wesley, Reading MA, 1988.

[Verr 90] A. Verroust, "Etude de Problèmes Liés a la Définition, la Visualisation et l'Animation d'Objets Complexes en Informatique Graphique", thèse d'Etat, Universite de Paris-sud, 1990.

[Wils 91] M.Wilson & A. Conway, "Enhanced Interaction Styles for User Interfaces", IEEE Computer Graphics & Applications, pp 79-90, Mars 1991.

ANNEXE A

- **Description des boutons de la zone de commande**
- **Les fonctions de configuration d'interface**

Description des boutons de la zone de commande

Nous décrivons ci-dessous les boutons et les menus de la zone de commande.

1- DONE
Une fois la saisie terminée, l'utilisateur doit informer le système en sélectionnant l'item "DONE". La déduction de la sémantique, à partir des structures de données des cartes planaires, est alors activée. Cet déduction se traduit par une mise à jour de la base de connaissances de l'application.

2- EDIT/DESIGN
Il existe deux modes dans le module OD: le mode d'édition "EDIT" où toute action de l'utilisateur produit une expression CMR qui sera envoyée au système; le mode de conception "DESIGN" où l'utilisateur peut saisir les plans du bâtiment et du réseau sans que le système en soit informé.

3-GESTURE/GRAPHIC
Cette commande permet de basculer entre le mode graphique et le mode gestuel. En mode graphique, les événements reçus du système de fenêtrage sont aiguillés vers le module OD alors qu'en mode gestuel les événements reçus sont aiguillés vers le module gestuel. A chaque mode est associé un curseur indiquant à l'utilisateur le mode dans lequel il se trouve.

4- CLEAR
Ce bouton permet d'effacer ce qui est affiché sur l'écran, par exemple le plan d'un étage sans pour autant détruire l'étage courant. Cela permet à l'utilisateur de saisir de nouveau le plan de l'étage ou du réseau ou les deux à la fois.

5-DELETE
A l'inverse du bouton "CLEAR", le bouton "DELETE" détruit entièrement l'étage courant avec son réseau.

6-LOAD
Le bouton "LOAD" possède un menu contenant la liste de tous les étage du bâtiment. La sélection d'un étage conduit à l'affichage de ce dernier, qui devient l'étage courant.

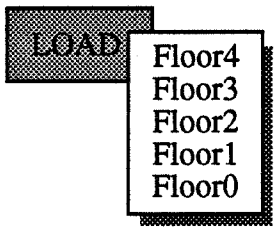
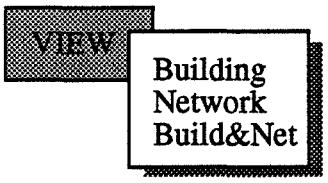


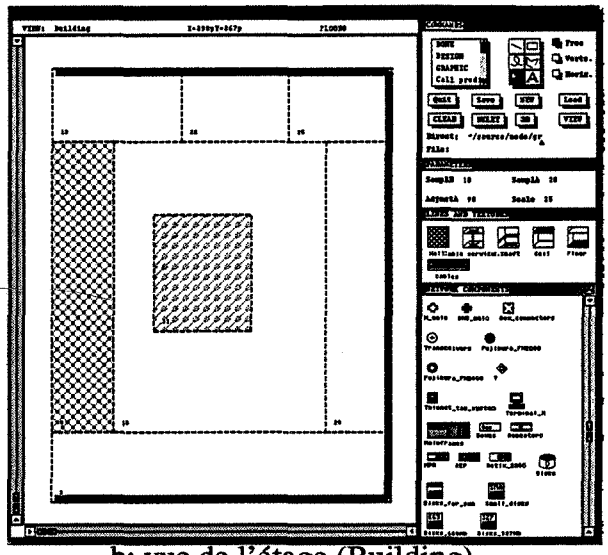
Figure 70: Le bouton "LOAD"

7-NEW
La sélection du bouton "NEW" permet de créer un nouvel étage (qui sera l'étage situé juste au dessus de l'étage courant).

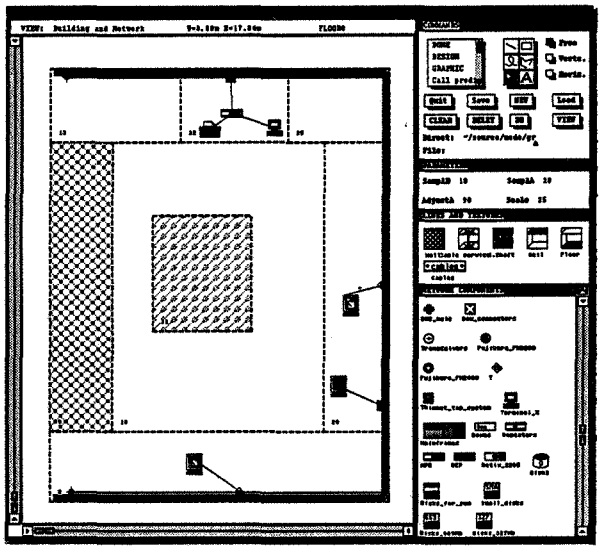
8-VIEW (voir la figure 71)
Ce bouton permet d'afficher soit uniquement l'étage courant du bâtiment, soit uniquement le réseau de l'étage courant, soit l'étage courant avec son réseau.



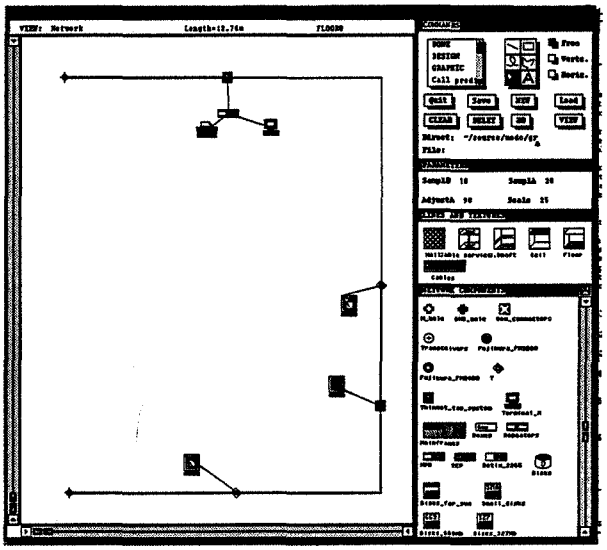
a: le bouton "VIEW"



b: vue de l'étage (Building)



c: vue de l'étage et de son réseau (Build&Net)



d: vue du réseau de l'étage courant (Network)

Figure 71: Le bouton view

9- QUIT
Ce bouton permet de détruire les trois fenêtres du module OD en informant le système de l'action de l'utilisateur.

10- 3D
La fenêtre 3D permet une représentation graphique des objets donnant l'impression d'une représentation en trois dimensions. Outre la représentation graphique, la fenêtre 3D permet certaines manipulations d'objets qui ne peuvent être accomplies que sur une telle représentation; par exemple, l'installation d'une gaine verticale qui marque le point de passage d'un câble vertical dans l'étage.

Voici une description sommaire des opérations qui peuvent être activées à partir d'un menu de la fenêtre 3D:

PUT VERTICAL SHAFT:
Après avoir sélectionné deux points sur deux étages différents (par appui sur le bouton du milieu de la souris), l'activation l'item INSTALL VERTICAL CABLE crée une gaine verticale. Cette gaine verticale peut être de trois types: soit de type escalier (stair), soit de type ascenseur (Lift) soit de type "puit". Le type de la gaine verticale est défini préalablement par un menu de la zone de commande.

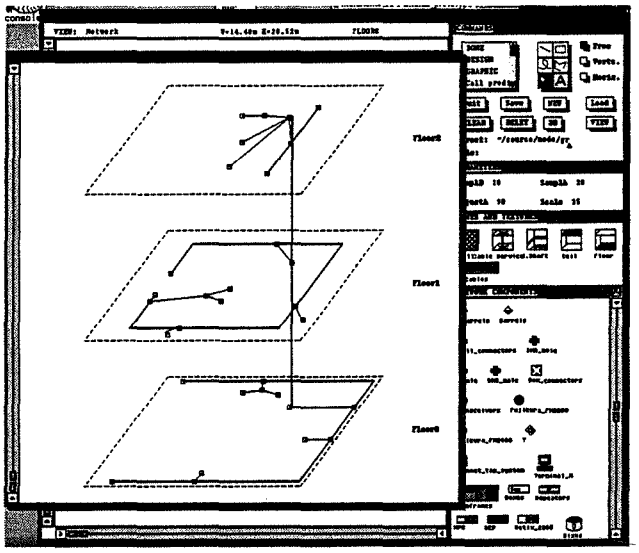
ALIGN SHAFT:
Après sélection de toutes les icônes représentant la gaine verticale, cette option permet d'aligner verticalement la gaine.

PUT VERTICAL CABLE
Après sélection de toutes les icônes représentant la gaine verticale, cette option permet d'installer un câble vertical joignant tous les étages contenant une gaine verticale sélectionné.

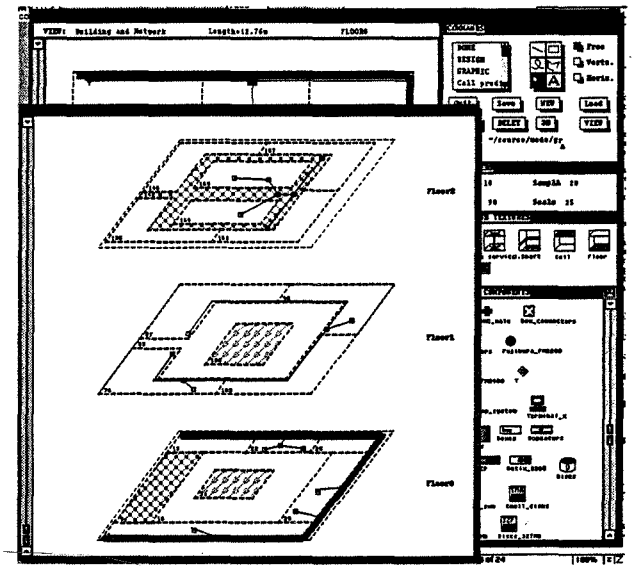
DELETE VERTICAL SHAFT
Après sélection de la gaine verticale, cette option permet la destruction de celle-ci.

DELETE VERTICAL SHAFT
Après sélection du câble vertical, cette option permet la destruction de celui-ci.

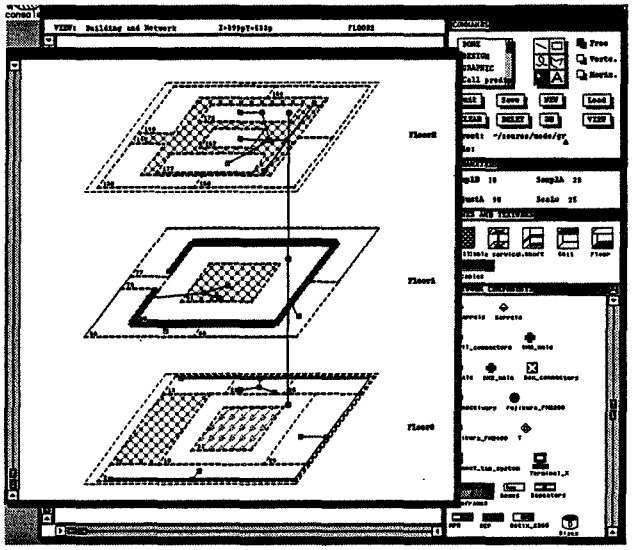
Notons que les représentations 2D et 3D utilisent les mêmes structures de données. En effet, il n' y a pas de duplication des structures de données pour éviter l'encombrement de la mémoire et assurer la cohérence de la représentation. Ainsi toute modification prise en compte est visualisée simultanément en 2D et en 3D.



Le réseau



Le batiment



Le réseau et le bâtiment

Figure 72: Différentes vues en 3D

Les fonctions de configuration d'interface

emse_create_zone(char, ident, cursor_image,x,y,w,h,window)

char: *chaîne de caractère contenant l'entête de la zone, si cette chaîne est vide la zone sera sans entête.*

ident: *identificateur de la zone (constante préalablement définie)*

cursor_image: *un tableau décrivant l'image du curseur quand il est à l'intérieur de la zone*

x,y,w,h: *des entiers précisant la position, la largeur et la longueur de la zone*

window: *identificateur de la fenêtre contenant la zone*

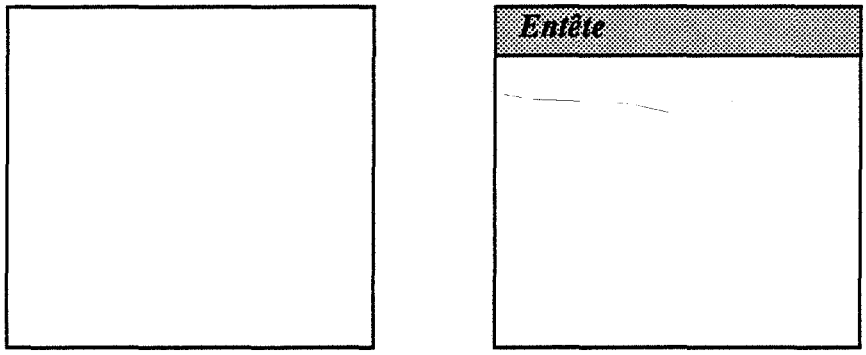


Figure 73: Les différents types de zone

emse_create_item(type, ident, text, image, menu, x,y, function, window,w,h,flag,shad)

type: *type de l'item (bouton, message, item texte)*

ident: *identificateur de l'item*

text: *si l'item est un bouton,* ***text*** *sera le texte affiché dans le bouton. Si le bouton contient une image,* ***text*** *sera affiché en dessous de l'image. Si l'item est un item texte,* ***text*** *sera la partie fixe de l'item*

image: *un tableau décrivant l'image de l'item*

menu: *un tableau contenant le menu associé au bouton. Ce menu se "déroule" si on appuie sur le bouton droit de la souris en étant positionné sur l'item*

x,y: *position de l'item*

function: *le nom de la fonction qui sera activée quand on appuie sur l'item*

window: *identificateur de la fenêtre contenant l'item*

w,h: *la longueur et la largeur de l'item*

flag: *c'est une variable spéciale indiquant que l'activation de l'item implique la déactivation de tous les autres items ayant le même flag*

shad: *variable indiquant que l'item est "ombré"*

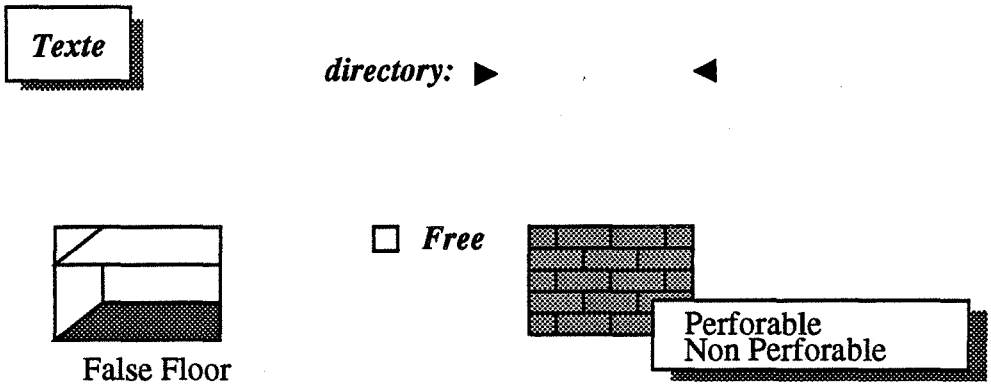


Figure 74:Quelques types d'item

emse_create_scroll(position, x,y,w,h,function,window)
position: *position de l'ascenseur (nord,sud, est, ouest)*
x,y,w,h: *les coordonnées de l'ascenseur*
function: *la fonction qui sera activée lorsqu'on déplace l'ascenseur,*
window: *identificateur de la fenêtre*

emse_create_item_menu(type, ident, menu,x,y,fonctions,window)
type: *type du menu (menu fixe,menu déroulant, menu fixe avec ascenseur)*
ident: *identificateur du menu*
x,y: *position du menu*
fonctions: *liste des fonctions associées à chaque item du menu*
window: *identificateur de la fenêtre*

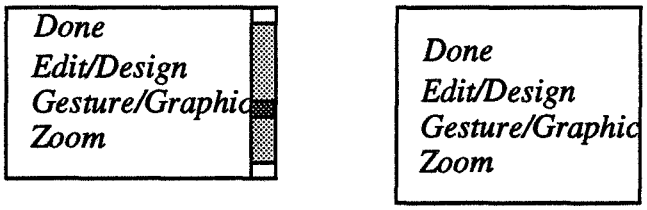


Figure 75: Les différents types de menu

emse_create_canvas(ident,x,y,w,h)
ident: *identificateur de la fenêtre*
x,y,w,h: *coordonnées de la fenêtre*

emse_init_frame(ident,fr,x,y,w,h,function)
ident: *identificateur du frame*
x,y,w,h: *coordonnées de la fenêtre*
function: *fonction activée avant la fermeture de la fenêtre*

emse_create_bitmap(ident, x,y,w,h,windows)
ident: *identificateur du bitmap*
x,y,w,h: *coordonnées du bitmap*
window: *identificateur de la fenêtre associée*

ANNEXE B

- Liste des prédicats permettant de communiquer avec l'outil de dessin

Liste des prédicats permettant de communiquer avec l'outil de dessin

En mode entrée

gr_net_u_select/1
arg1: identificateur de l'objet sélectionné

Quand l'utilisateur sélectionne un objet de l'OD, un message CMR doit informer le système pour qu'il mette à jour le contexte de dialogue. En effet, l'objet sélectionné peut être désigné par l'intermédiaire de la langue naturelle.

gr_net_u_move/4
arg1: identificateur d'un objet (par exemple sun48)
arg2: identificateur de l'ancienne pièce (par exemple room1)
arg3: identificateur de la nouvelle pièce (par exemple room2)
arg4: variable d'autorisation (par exemple OK/ERROR)

Quand l'utilisateur déplace un objet, le système est informé de cette action. Si l'action est autorisée, l'argument 4 reçoit la valeur OK et l'OD valide l'action; sinon, l'argument 4 reçoit la valeur ERROR et l'OD remet l'objet à son ancienne place.

gr_net_u_create/3
arg1: identificateur de l'objet créé (par exemple disk1)
arg2: type de l'objet (par exemple Small_disks)
arg3: variable d'autorisation

Par ce prédicat, l'OD informe le système que l'utilisateur veut créer un objet. Le type de l'objet est défini par arg2. Si le système autorise la création de l'objet, l'argument 3 reçoit la valeur OK; sinon il reçoit la valeur ERROR et l'objet ne sera pas inséré dans la liste des objets de l'OD.

gr_net_u_locate_object/3
arg1: identificateur de l'objet (par exemple machine1)
arg2: identificateur d'une pièce (par exemple room3)
arg3: variable d'autorisation

Quand l'utilisateur crée un objet, il le place dans une pièce. L'OD localise l'objet et informe le système Celui-ci répond par OK ou ERROR, selon que l'information est cohérente ou non.

gr_net_u_delate/2
arg1: identificateur de l'objet (par exemple machine1)
arg2: variable d'autorisation

Ce prédicat informe le système que l'utilisateur veut détruire un objet. Si la

destruction est autorisée, l'argument 2 prendra la valeur OK et l'objet sera détruit dans l'OD et dans le système; sinon l'action est refusée et l'objet ne sera pas détruit

gr_net_u_link/3
arg1: identificateur d'un objet (par exemple machine1)
arg2: identificateur d'un objet (par exemple Disk1)
arg3: variable d'autorisation

Ce prédicat demande au système l'autorisation de connecter deux objets. Si l'autorisation est acceptée (arg3=OK), les deux objets sont connectés; sinon (arg3=ERROR), les deux objets ne seront pas connectés.

gr_net_u_unlink/3
arg1: identificateur d'un objet (par exemple machine1)
arg2: identificateur d'un objet (par exemple Disk1)
arg3: variable d'autorisation

Ce prédicat demande au système l'autorisation de déconnecter deux objets. Si l'autorisation est acceptée (arg3=OK), les deux objets sont déconnectés; sinon (arg3=ERROR), les deux objets restent connectés.

gr_net_u_reclassify/4
arg1: identificateur de l'objet (par exemple machine1)
arg2: ancienne classe de l'objet (par exemple Sun3)
arg3: nouvelle classe de l'objet (par exemple Sparcs)
arg4: variable d'autorisation

Ce prédicat informe le système que l'utilisateur veut changer la classe d'un objet. Si ce changement est autorisé (arg4=OK) l'OD exécute cette opération; sinon (arg4=ERROR), l'objet ne changera pas de classe

gr_net_u_set_feature/3
arg1: identificateur de l'objet (par exemple shaft3)
arg2: attribut de l'objet (par exemple wall_succesion)
arg3: nouvelle valeur de l'attribut (par exemple [wall1,wall2,wal5,wall6])

Quand l'utilisateur manipule les objets graphiques, il provoque souvent la modification de leurs attributs. Ce prédicat permet au système de modifier les attributs des objets dans la base de connaissances.

gr_net_get_var/2
arg1: variable de sortie qui sera initialisée par le système et contiendra un identificateur pour un objet qui vient d'être crée.
arg1: type de l'objet (par exemple Room)

L'OD utilise ce prédicat pour demander au système de donner un identificateur à un

objet.

gr_net_u_quit/0

Ce prédicat informe le système que l'utilisateur veut "quitter" l'outil OD.

Quand l'utilisateur se met en mode gestuel, l'OD envoie les événements qu'il reçoit du système de fenêtrage en utilisant le prédicat suivant:

gr_net_handle_network_event/3
arg1: le bouton de la souris activé (par exemple MS_LEFT/MS_MIDDLE/ MS_DRAG)
arg2: un pointeur sur la fenêtre où se déroule l'action
arg3: un pointeur sur l'événement

En mode sortie

gr_net_s_highlight/1
arg1: identificateur d'un objet

Ce prédicat appelle une fonction C qui affiche un objet en inverse vidéo. Le système active ce prédicat pour désigner un objet à l'utilisateur.

gr_net_s_move/2
arg1: identificateur d'un objet
arg2: identificateur de la pièce.

Ce prédicat est activé par le système pour déplacer un objet d'une pièce à une autre. Ceci ne concerne que les objets dont la représentation graphique est une icône. L'emplacement de l'objet dans la pièce est aléatoire, c'est-à-dire que la fonction C activée par ce prédicat doit déterminer la position de l'objet dans la pièce de façon aléatoire.

gr_net_s_create/2
arg1: identificateur d'un objet à créer
arg2: type de l'objet à créer

Ce prédicat est utilisé par le système pour créer un objet graphiquement. Ceci concerne les objets tels que les câbles et l'équipement matériel du réseau; les autres objets ne peuvent pas être créés graphiquement car l'application ne dispose pas d'informations géométriques sur les objets.

gr_net_s_locate/2
arg1: identificateur d'un objet
arg2: identificateur d'une pièce.

Ce prédicat permet au système, après avoir créé un objet, de le placer dans une pièce. Comme le prédicat gr_net_s_move, l'emplacement de l'objet dans la pièce est aléatoire.

gr_net_s_delete/1
arg1: identificateur d'un objet

Ce prédicat permet au système de détruire un objet. Si l'objet est connecté à d'autre objets, toutes les connexions seront détruites.

gr_net_s_link/2
arg1: identificateur d'un objet
arg2: identificateur d'un objet

Ce prédicat permet de connecter deux objets en créant un lien entre eux.

gr_net_s_unlink/2
arg1: identificateur d'un objet
arg2: identificateur d'un objet

Ce prédicat permet au système de détruire la connexion entre les deux objets désignés par leur identificateur.

gr_net_s_reclassify/3
arg1: identificateur d'un objet
arg2: ancien type de l'objet
arg3: nouveau type de l'objet

Ce prédicat permet au système de reclassifier un objet.

gr_net_s_clear_network/0

Ce prédicat permet au système de détruire tous les objets du réseau.

gr_net_s_clear_building/0

Ce prédicat permet au système de détruire tous les objets du bâtiment.

gr_net_s_get_object/3
arg1: identificateur d'un objet; valeur retournée
arg2: entier, abscisse du point
arg3: entier, ordonnée du point

Ce prédicat retourne l'identificateur de l'objet qui contient le point (arg2,agr3). Ce prédicat est utilisé essentiellement pendant le mode gestuel pour identifier l'objet concerné par le geste de l'utilisateur.

gr_net_s_get_object_list/1
arg1: liste des objets créés

Ce prédicat retourne la liste des objets créés dans l'OD.

gr_net_s_get_visible_object_list/1
arg1: liste des objets visibles

Ce prédicat retourne la liste des objets visibles dans la fenêtre 2D.

Il est nécessaire que le système puisse effectuer un raisonnement spatial. Par exemple, pour permettre au système de savoir si une machine est à gauche ou à droite d'un disque. Ce raisonnement est basé sur les coordonnées des objets. On peut définir ainsi les prédicats nécessaires pour un tel raisonnement:

gr_net_s_left_of/2
arg1, arg2: identificateur d'objet

Ce prédicat rend la valeur VRAI si arg1 est à gauche de arg2.

gr_net_s_right_of/2
arg1, arg2: identificateur d'objet

Ce prédicat rend la valeur VRAI si arg1 est à droite de arg2.

gr_net_s_above/2
arg1, arg2: identificateur d'objet

Ce prédicat rend la valeur VRAI si arg1 est au dessus de arg2 dans un même étage ou si arg1 est situé dans un étage supérieur à l'étage où se trouve arg2.

gr_net_s_below/2
arg1, arg2: identificateur d'objet

ce prédicat rend la valeur VRAI si arg1 est au dessous de arg2 dans un même étage ou si arg1 est situé dans un étage inférieur à l'étage où se trouve arg2.

gr_net_s_front_of/2
arg1, arg2: identificateur d'objet

Ce prédicat rend la valeur VRAI si l'objet arg1 cache, même localement, l'objet arg2.

gr_net_s_behind_of/2

arg1, arg2: identificateur d'objet

Ce prédicat rend la valeur VRAI si l'objet arg1 est caché, même localement par l'objet arg2.

gr_net_s_get_features/3
arg1: identificateur de l'objet
arg2: nom de l'attribut
agr3: valeur de l'attribut

Ce prédicat permet au système de modifier la valeur d'un attribut de l'objet.

gr_net_s_up_OD_tool/0

Ce prédicat permet de lancer l'outil OD.

ANNEXE C

- Définition syntaxique de la CMR

Définition syntaxique de la CMR

La grammaire BNF de la CMR

Dans cette section, nous présentons la grammaire BNF de la CMR, cela permet de donner à la CMR un cadre indépendant de son implémentation en Prolog. Voici quelques conventions:

A = zéro ou plusieurs occurrences de A*
A+ = une ou plusieurs occurrences de A
A|B= A ou B

La CMR contient une liste d'expressions CMR plus d'autres informations telles que l'état, le mode, la date de l'action, les présuppositions de l'utilisateur, les erreurs de l'utilisateur et des informations syntaxiques. Ces informations ne sont pas définies sous forme BNF.

<CMR>::= <CMR_exp> <status> <mode> <time> <presupositions> <mistakes> <syntatic_info>*

<CMR_exp>::= <c_force> <annotation> <formula>*
*<c_force>::= <var> <force>**

les c_force sont divisées en groupes:

forces générales:
<gen_force>::= request|supply

forces spécifiques:
<spec_force>::= affirm| agree| confirm| deny| greeting| noncommit| noact| nonclass| hypothesis
sous-forces:
<sub_force>::= action| inform| referent| explain| open| close| cut

les c_forces interagissent de la manière suivante:
c_force::= <gen_force> <spec_force>
c_force::= <gen_force> <spec_force> <spec_force>
c_force::= <gen_force> <spec_force> <sub_force>
c_force::= <gen_force> <sub_force>

Les annotations peuvent représenter des aspects pragmatiques qui ne font pas partie de la forme logique, du contenu ou des forces de communication. Elles ont la forme

suivante:

*<annotation>::= <var> <anno>**
<anno>::= <var> <annotation_values>
<annotation_values>::=annotation_value <annotation_values>

Une formule est basée sur la logique de premièr ordre avec quelques extensions: le quantificateur de cardinalité, l'opérateur THE.

<formula>::= neg (<formula>)
<formula>::= and (<formula> <formula>+)
<formula>::= or (<formula> <formula>+)
<formula>::= cond (<formula> <formula>)
<formula>::= <descriptor> <formula>
<formula>::= <term> = <term>
<formula>::= named (<var>, <typed_constant>)
<formula>::= card (<typed_constant>+, integer)
<formula>::= <cases>

Un descripteur est un quantificateur et un type sémantique, ce qui permet de typer les variables quantifiées. On peut aussi limiter le type des variables en définissant une formule optionnelle appelée "restrictice".

<descriptor>::= <quantifier> <var> <type> [<formula>]
<quantifier>::= some| all| the| <cardinality>
<cardinality>::= exactly <integer>
<cardinality>::= at_most <integer>
<cardinality>::= at_least <integer>
<cardinality>::= more than<integer>
<cardinality>::= less_than <integer>
<cases>::= <var> <case>+
<case>::= <slot> <term>
<term>::= <var>| <typed_constant>
<typed_constant>::= <type> <constant>
<constant>::= un "atom" ou un nombre Prolog
<var>::= x + un nombre Prolog
<integer>::= un nombre Prolog
<type>::= un "atom" Prolog (dépendant de l'application)
<slot>::= un atom Prolog (dépendant de l'application)

Implémentation Prolog de la CMR

Considérons la phrase suivante:

qui dirige MMI2?

cette phase produit l'expression CMR suivante:

CMR_exp(	
c_force (x1,[request,referent])	acte de communication
[anno(x1,[indefinite,single]),	annotation sur quelqu'un
anno(x2,[definite,single]),	annotation sur MMI2
anno(x3,[present,non_perfective,fact])],	annotation sur "diriger"
[desc(some,x1,person,true),	descripteur pour quelqu'un
desc(some,x2,project,	descripteur pour MMI2
[and ([x2=MMI2, named(x2,type (string, MMI2))]),	restricteur pour MMI2
desc(some,x3,diriger,true),	descripteur pour "diriger"
cases(x3,[case(agent,x1),case(object,x2)])])	cas de "diriger"

La syntaxe de la CMR en Prolog est la suivante:

TypedConstant= type (Constant,Constant)
Var = x + un nombre Prolog
Integer = un nombre Prolog
Constant = un "atom" ou un nombre Prolog

CMR= CMR(CMR_Expressions , Status, Mode, Time, Presupositions, Misatkes, SuntacticInfo)
CMR_Expressions= [CMR_Expression|CMR_Expressions]
CMR_Expressions = []

Expression CMR:

CMR_Expression = CMR_exp(CommunicationForce,Annotations,Formula)

Force de communication:

CommunicationForce= c_force(Var, [Force|Forces])

Forces= [Force|Forces]
Forces= []

Force= request
Force= supply
Force= agree
Force= confirm
Force= deny
Force= greeting
Force= noncommit
Force= noact
Force= nonclass

Force= hypothesis
Force= action
Force= inform
Force= referent
Force= explain
Force= open
Force= close
Force= cut

Annotations:

Annotations= [Annotation|Annotations]
Annotations= []

Annotation= anno (Var, AnnotationValues)
AnnotationValues = [AnnotationValue|AnnotationValues]
AnnotationValues= []

Terme:

Term = Var
Term= TypedConstant

Formule:

Formula= neg(Formula)
Formula= and ([Formula|Formulas])
Formula= or ([Formula|Formulas])
Formula= cond (Formula, Formula)
Formula= [Descriptor|Formula]
Formula= Term = Term
Formula= named (Var, TypedConstant)
Formula= card (TypedConstants, Integer)
Formula= cases
Formula= true
Formulas= [Formula|Formulas]
Formulas= []

TypedConstants = [TypeConstant|TypedConstants]
TypedConstants= []

Descripteur:

Descripteur = desc (Quantifier, Var, Constant, Formula)

Quantifier = some

Quantifier = all
Quantifier = the
Quantifier = card (Comparative, Integer)

Comparative = exactly
Comparative = at_most
Comparative = at_least
Comparative = more_than
Comparative = less_than

Cas:

cases= cases (Var, [CaseValue\|CaseValues])

CaseValues= [CaseValue\|CaseValues]
CaseValues= []

CaseValue= case (Constant,Term)

Résumé :

L'objet de cette thèse est la construction d'un outil graphique dans le cadre du projet MMI2 (*A Multi-Mode Interface for Man-Machine Interaction with Knowledge based systems*). MMI2 est un projet Esprit II qui a pour objectif principal de construire une interface homme-machine intelligente intégrant plusieurs modes de communication: langue naturelle (anglais, français, espagnol), langage de commande, graphique et gestuel. Ce travail est le résultat d'une approche expérimentale de développement d'un outil de manipulation directe, batisé Outil de Dessin, dans un environnement multi-mode. Dans ce contexte, notre démarche a consisté à utiliser des techniques de construction géométrique dans la conception d'une interface. Nous proposons pour cela un modèle fondé sur le concept de carte planaire. Il conduit à des structures de données de type représentation par frontières dont la notion de base est le brin, qui est une arête topologique orientée. Ce modèle permet de proposer un algorithme de saisie et de modification de cartes planaires qui, tout en contrôlant à chaque instant la planarité de la carte, est capable de détecter automatiquement la fermeture des contours. Par ailleurs, nous nous sommes intéressés à l'étude de la sémantique graphique, pour permettre l'intégration des différents modes de communication pour assurer un dialogue multi-mode entre l'utilisateur et l'ordinateur.

Mots-clés

Interface homme-machine, ergonomie, carte planaire, gestuel, interface multi-mode.

Abstract:

The aim of this thesis is the construction of a graphic tool as part of the MMI2 project (*A Multi-Mode Interface for Man-Machine Interaction with Knowledge based systems*). MMI2 is an Esprit II project whose main purpose is to construct an intelligent man-machine interface which includes several communication modes: natural language (english, french, spanish), command language graphics and gesture. This work is the result of an experimental approach for the construction of a direct manipulation tool, named Drawing tool, in a multi-modal environment. In this context, our approach consisted in using a geometric construction models in the conception of the interface. We suggest a model based on the planar map concept. It involves polygonal data structures the elementary component of which is half-oriented edges called "brins". We started developping an algorithm to capture and to modify the planar maps which control, at any moment, the planarity of the maps, and automatically detect the polygons closness. Furthermore, we studied the semantic of graphics taking into account the integration of the different communication modes to assure a multi-modal dialog between user and computer.

Key-words

Man-machine interface, ergonomic, planar map, gesture, multi-modal interface.